

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 06-274333

(43)Date of publication of application : 30.09.1994

(51)Int.Cl.

G06F 9/06

G06F 9/06

G06F 3/14

(21)Application number : 05-061603

(71)Applicant : HITACHI LTD

(22)Date of filing : 22.03.1993

(72)Inventor : OSHIMA YOSHIMITSU

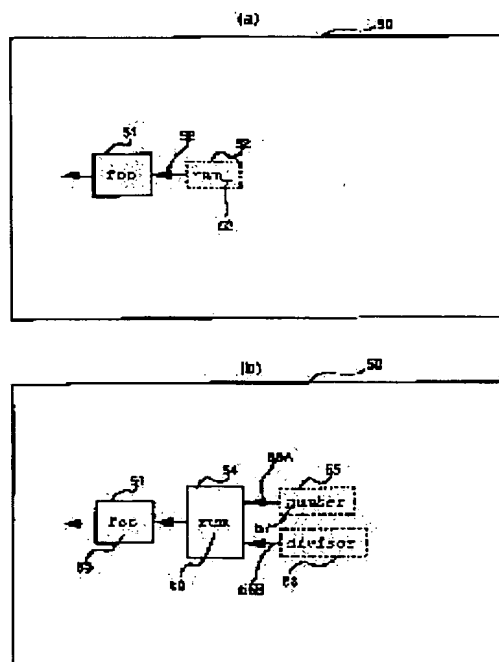
(54) INPUT SUPPORT METHOD AND EDITING SUPPORT METHOD FOR PROGRAM

(57)Abstract:

PURPOSE: To input a program expressed by a graphic only with a keyboard by displaying the graphic showing a constitution element in response to a character string showing the constitution element of the program and displaying an area to be inputted near the graphic when the other constitution element which is to be inputted in relation to the constitution element exists.

CONSTITUTION: It is assumed that the character string rem is inputted to a character string input area 52, for example. When a conversion key on the keyboard is depressed, the character string input area 52 is disappeared, and a template by the graphic as against a function rem, namely, the graphic 54, the character string input areas 55 and 56 for the two arguments and arrows 55A and 55B connecting the areas to the graphic 54 are displayed.

Temporary argument names are respectively displayed in the character string input areas 55 and 56, and therefore a user can judge what is to be inputted next. When input from the keyboard is advanced in such a state, the temporary argument name in the character string input area 55 is deleted, and it is changed into the character string inputted from the keyboard.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

Copyright (C); 1998,2003 Japan Patent Office

(19)日本国特許庁(JP)

(12)公開特許公報(A)

(11)特許出願公開番号

特開平6-274333

(43)公開日 平成6年(1994)9月30日

(51)Int.Cl.⁵

G 0 6 F 9/06

3/14

識別記号

4 3 0 P 9367-5B

4 4 0 B 9367-5B

3 1 0 E 7165-5B

庁内整理番号

F I

技術表示箇所

審査請求 未請求 請求項の数33 O L (全 36 頁)

(21)出願番号 特願平5-61603

(22)出願日 平成5年(1993)3月22日

(71)出願人 000005108

株式会社日立製作所

東京都千代田区神田駿河台四丁目6番地

(72)発明者 大島 義光

東京都国分寺市東恋ヶ窪1丁目280番地

株式会社日立製作所中央研究所内

(74)代理人 弁理士 小川 勝男

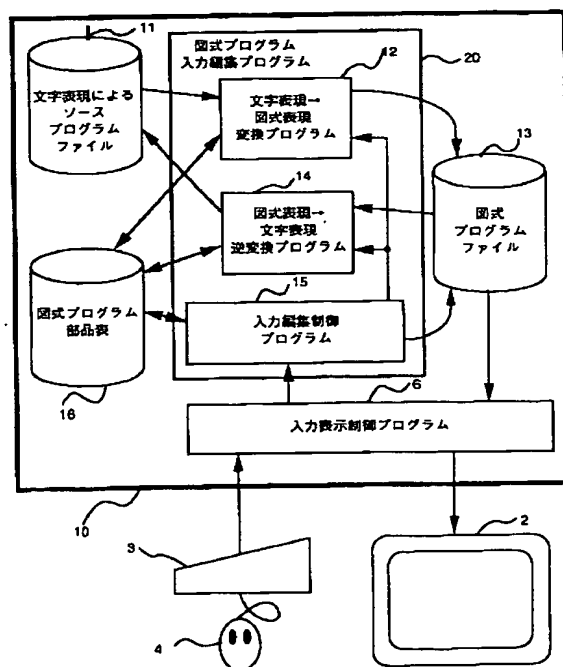
(54)【発明の名称】 プログラムの入力支援方法および編集支援方法

(57)【要約】 (修正有)

【目的】 プログラムのキーボードによる入力を支援し、あるいは入力済みのプログラムの編集を支援する。

【構成】 プログラム構成要素、例えば、関数名称fooが入力されると、その名称を内部に含む図形51とともに、その関数が必要とする引数を入力する領域52を矢印58で図形51に連結して表示し、この領域の先頭に文字カーソル53を自動的に移動する。この領域に引数として、他の関数の名称remが入力されると、この関数名称を内部に含む図形54とそれが必要とする二つの引数を入力する領域55、53とそれらを図形54に連結する矢印55A、55Bを表示する。このような操作を繰り返して、一連のプログラム構成要素の名称の入力をしやすくする。また、表示されている図形に対して、挿入、ペースト、移動、複写などの編集が指定されたときに、選択された図形の中でその編集を実行可能な部分を表示して、図形的編集を支援する。

(図1)



【特許請求の範囲】

【請求項1】表示画面上のいずれかの位置に入力された、入力すべきプログラムを構成するいずれかの構成要素を表す文字列に応答して、その文字列に対応して予め記憶された、その構成要素を表す図形をその位置に表示し、その構成要素に関連して入力すべき少なくとも一つの他の構成要素があるときには、その、他の構成要素を入力すべき領域を、その図形の近傍に表示するプログラム入力支援方法。

【請求項2】該領域を、上記図形に対して定められた所定の相対的位置関係を有する位置に表示するプログラム請求項1記載の入力支援方法。

【請求項3】該領域を、上記図形に連結して表示するプログラム請求項1記載の入力支援方法。

【請求項4】上記構成要素が引数を必要とする場合に、その引数を入力すべき領域を、上記他の構成要素を入力する領域として表示する請求項1記載のプログラム入力支援方法。

【請求項5】上記入力すべき他の構成要素の種別を表す文字列を表示するステップをさらに有する請求項1記載のプログラム入力支援方法。

【請求項6】上記構成要素が引数を必要とする場合に、上記領域は、その引数を入力すべき領域であり、上記他の構成要素の種別を表す文字列は、その構成要素が有する仮引数の名称である請求項5記載のプログラム入力支援方法。

【請求項7】上記他の構成要素の種別を表す文字列は、上記領域の内部に表示される請求項5記載のプログラム入力支援方法。

【請求項8】上記他の構成要素の種別を表す文字列は、上記領域の内部に表示される請求項6記載のプログラム入力支援方法。

【請求項9】上記他の構成要素の種別を表す文字列は、上記図形の内部またはその近傍の、上記領域に近い位置に表示される請求項9記載のプログラム入力支援方法。

【請求項10】上記領域を表示後、文字入力位置を表すカーソルを、上記領域内の先頭位置に移動するステップをさらに有する請求項1記載のプログラム入力支援方法。

【請求項11】上記構成要素がその構成要素に関連して入力すべき他の構成要素を必要としない場合には、その図形の後方または上位で一番近い図形が表す構成要素に対して表示された、それが必要とする他の構成要素を入力するための領域の先頭位置に、上記文字カーソルを移動するステップをさらに有する請求項10記載のプログラム入力支援方法。

【請求項12】上記他の構成要素として、その、他の構成要素を形成するプログラム文が入力されたとき、そのプログラム文の構文を解析し、

その構文を反映した一群の図形を、上記領域に代えて、かつ、上記図形に対して所定の相対的位置関係を有する位置に表示するステップをさらに有する請求項1記載のプログラム入力支援方法。

【請求項13】該表示された領域に該他の構成要素を表す文字列の一部またはその文字列が入力されたことに応答して、該領域内に表示されている、該他の構成要素の種類を表す文字列を消去し、該入力された文字列を該領域内に表示するステップをさらに有する請求項7記載のプログラム入力支援方法。

【請求項14】該入力された構成要素が、複数の他の構成要素を必要とする場合、上記表示のときに、それぞれ該複数の他の構成要素を入力するための複数の領域を、上記図形の近傍に表示する請求項1記載のプログラム入力支援方法。

【請求項15】該複数の領域は同時に表示される請求項14記載のプログラム入力支援方法。

【請求項16】該入力された構成要素が、同一種類の複数の他の構成要素を必要とするがその数は任意である場合、上記表示のときに、該複数の領域は順次表示され、その順次表示は、該複数の領域の一つを表示し、その表示された一つの領域に他の構成要素を表す文字列が入力された後で、該複数の領域の他の一つの領域を表示するごとく行なう請求項14記載のプログラム入力支援方法。

【請求項17】該複数の領域は、その内の先行して表示された領域の下方に表示される請求項16記載のプログラム入力支援方法。

【請求項18】該他の構成要素が、省略可能な構成要素である場合、ユーザ指示に応答してその領域を消去するステップをさらに有する請求項1記載のプログラム入力支援方法。

【請求項19】該消去は、その領域に対して、未だ文字列が入力がなされていないか、もしくは空白文字しか入力がなされていないときで、かつ、その領域中に文字カーソルがあるときに、ユーザによる特定のキーまたはボタンの押下を契機として、行なわれる請求項18記載のプログラム入力支援方法。

【請求項20】ユーザによりプログラムの編集終了が後に指示されたとき、上記他の構成要素が省略不可能な構成要素であり、かつ、上記領域に対して、未だ文字列が入力がなされていないか、もしくは空白文字しか入力がなされていない状態にあるかを判別し、判別の結果が肯定的である場合、この領域への文字入力が終了していないことが識別可能なように、その領域の表示を変えるステップをさらに有する請求項1記載のプログラム入力支援方法。

【請求項21】上記領域に文字列により入力したプログラムに一つ以上の不足の構成要素があるとき、その構成要素に対応する入力領域を該当位置に含めた形で、その

プログラムに対応する図形を表示する請求項1記載のプログラム入力支援方法。

【請求項22】上記領域に文字列により入力したプログラムに一つ以上の過剰な構成要素があるとき、その構成要素を識別表示する形で、そのプログラムに対応する図形を表示する請求項1記載のプログラム入力支援方法。

【請求項23】表示画面上のいずれかの位置に入力された、入力すべきプログラムを構成するいずれかの構成要素を表す文字列に回答して、その文字列に対応して予め記憶された、その構成要素を表す図形をその位置に表示し、

その構成要素に関連して入力すべき少なくとも一つの他の構成要素があるときには、その、他の構成要素を入力すべき、その図形の近傍の、上記構成要素により定まる位置を該表示画面上に指示するプログラム入力支援方法。

【請求項24】上記指示は、文字カーソルをその位置に移動させて行なわれる請求項23記載のプログラム入力支援方法。

【請求項25】表示画面上のいずれかの位置に入力された、入力すべきプログラムを構成するいずれかの構成要素を表す文字列に回答して、その文字列に対応して予め記憶された、その構成要素を表す図形をその位置に表示し、

上記表示ステップを、それぞれ他の構成要素を表す複数の文字列に回答して繰り返し、もって複数の図形を表示し、

その際、いずれかの先行して入力された構成要素が必要とする後続の構成要素を表す図形を表示する位置は、その先行して入力された構成要素を表す図形の位置に依存して定められるプログラム入力支援方法。

【請求項26】表示画面上のいずれかの位置に入力された、入力すべきプログラムを構成するいずれかの構成要素を表す文字列に回答して、その文字列に対応して予め記憶された、その構成要素を表す図形をその位置に表示し、

上記構成要素に関連して入力すべき他の構成要素として、その、他の名称を表す文字列が入力されたとき、その文字列に対して定められた他の図形をその図形に連結し表示し、

上記他の構成要素としてプログラム文が入力されたとき、そのプログラム文の構文を解析し、その構文を反映した一群の図形を、上記図形に対して所定の位置関係を有する位置に表示するプログラム入力支援方法。

【請求項27】表示画面上のいずれかの位置に入力された、入力すべきプログラムを構成するいずれかの構成要素を表す文字列に回答して、その文字列に対応して予め記憶された、その構成要素を表す図形をその位置に表示し、

上記表示ステップを、それぞれ他の構成要素を表す複数の文字列に回答して繰り返し、もって複数の図形を表示し、

表示された複数の図形の少なくとも一つに関して指示された編集態様に回答して、その一つの図形が、新たな構成要素を表す図形になるように、その図形をその編集指示で指定される編集態様に従って編集するプログラム編集支援方法。

【請求項28】挿入コマンド発行後、図形の形式で表示したプログラムの上で、新たな構成要素を挿入可能な位置に図形カーソルを位置付けたとき、プログラムを表わす図形の該当部分を識別表示する請求項27記載のプログラム編集支援方法。

【請求項29】挿入コマンド発行後、図形の形式で表示したプログラムの上で、新たな構成要素を挿入不可の位置に図形カーソルを位置付け、挿入実行を指示する特定のキーまたはボタンを押下したとき、警告音、警告メッセージ等の警告情報を提示する請求項27記載のプログラム編集支援方法。

【請求項30】挿入コマンド発行後、図形の形式で表示したプログラムの上で、新たな構成要素を挿入可能な位置に図形カーソルを位置付け、挿入実行を指示する特定のキーまたはボタンを押下したとき、その挿入位置によって決定される構成要素の範疇に対応する形で表現した文字列入力領域を、その位置に挿入する請求項27記載のプログラム編集支援方法。

【請求項31】図形で表示したプログラムの構成要素が、その一部にその構成要素を指示する名称またはキーワードを含む場合、その名称またはキーワードを表わす文字列を、別の構成要素を指示する文字列に変更し、構成要素の変更を指示するコマンドの発行により、その構成要素を、変更後の文字列が表わす名称またはキーワードが指示する構成要素に対応する図形に変更する請求項27記載のプログラム編集支援方法。

【請求項32】文字形式への変換を指示するコマンドの発行により、図形形式で表示したプログラムのあらかじめ指定した部分を、対応する文字形式プログラムに変換し、元の図形形式プログラムと置き換え表示する請求項27記載のプログラム編集支援方法。

【請求項33】上記編集後の、表示された複数の図形を、それらが表す構成要素を連結したプログラムに変換するステップをさらに有する請求項27記載のプログラム入力支援方法。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は、図形を用いて表現したコンピュータ・プログラムの入力および編集方法に関する。

【0002】

【従来の技術】一般に図で表現した情報は理解しやす

い。もともとが2次元または3次元による形状を持つものは当然として、抽象的な情報も図で表現すると理解しやすくなることは、日常よく経験することである。コンピュータの世界においても、プログラムその他の情報を図で表現することにより、プログラム関連情報の理解を容易にする方法が、比較的古くから行なわれている。プログラムの図的表現に関して言えば、フローチャートがその嚆矢であり、その後、文献「プログラム制御構造の新しい表現法：木構造チャートが普及期に」（日経コンピュータ、1989. 1. 9、pp. 71～83）に紹介されているような改良案がいろいろと提案されている。最近では図による表現しか持たない、いわゆる視覚言語の研究も盛んである。

【0003】プログラムの図的表現およびそれに基づくプログラム開発ツールの一例として、文献「“SEWB”の開発思想と機能」（日立評論、Vol. 70、No. 2（1988. 2））で紹介されているプログラムの図式表現法PAD、およびそれに基づくプログラムのエディタ（PADエディタ）がある。PADは、主として、C、FORTRAN、COBOLなどの手続き型言語に共通する三つの制御構造—逐次実行、条件分岐、繰り返し—を図式化することによって、プログラムの構造を見やすくしようとするプログラムの図式化方法である。PADエディタは、このPADによるプログラム作成を支援する。

【0004】PADによるプログラムの記述例を図32に示す。図の（a）はFORTRANプログラムの例、（b）は（a）のPADによる表現である。図からわかるように、繰り返しdoは右端に縦棒を付加した箱により（同図21）、条件分岐IFは右側をへこました箱により（同図22）、また、通常の実行文は単純な長方形の箱により（同図23、24、25）、それぞれ表現している。このように、上記三つの制御構造を図で表わし、一方、詳細な処理はそれぞれの言語固有の文法にしたがって文字（テキスト）のまま表現する、というのがPADの基本的な表現法である。

【0005】PADエディタでプログラムを入力する場合、図形部分は、前記一般的な図の作成ツールと同様に、ディスプレイ画面上に表示した部品メニューから希望のものをマウスで選ぶことによって指定する。また、箱の中に記述する文字部分はキーボードから入力する、という方法をとっている。

【0006】ところで、図2からも容易にわかるように、プログラムの図式表現は、プログラムの構造、すなわち、構文構造を明示的に表わしている。したがって、プログラム言語の構文情報を利用して、プログラムの作成を支援するエディタを実現すれば有効であろう。具体的には、入力途中にプログラムの構文を示して入力进行を助けること、入力されたプログラムをチェックし、構文誤りがあるとき警告を発すること、また、構文誤りを発生

させるプログラムの入力を許さない、などの機能が考えられる。このようなエディタは、従来、構文主導型のエディタ（syntax directed editor）と呼ばれている。

【0007】テキスト・ベースのエディタについては、このような構文主導型のエディタがいくつか提案されており、その一例が文献T. Teitelbaum, et al ; The Cornell Program Synthesizer : A Syntax-directed Programming Environment, Communication of the ACM, Vol. 24, No. 9, pp. 563-573 (September 1981)に紹介されている。これは、言語要素の名称とそれを構成する引数の入力スロット（上記論文ではこれをplaceholderと呼んでいる）からなる構文テンプレートを準備し、この構文テンプレートを基本としてプログラムの入力編集を行なうようにすることにより、常に構文の正しい状態を維持しつつ、プログラムの入力編集を可能にするプログラム・エディタである。

【0008】また、テキスト・ベースの構文主導型のエディタにおける別の例として、先願（特開平1-118925）で述べているものがある。

【0009】これは、上記文献による構文主導型エディタが、常に正しい構文を維持しつつ入力編集を行なわせるという、ユーザにとってはある意味では非常に窮屈な点を緩和し、操作性を向上することを狙ったものである。具体的には、通常の文字単位に入力可能な画面エディタをベースに、構文テンプレートを挿入する機能を付加したものである。

【0010】図式による構文主導型のプログラム・エディタに関しては、一例が、文献 安井嗣了、南山智之「構文指向型PADエディタについて」（6S-1、情報処理学会第34回（昭和62年後期）全国大会）に紹介されている。

【0011】

【発明が解決しようとする課題】前記PADないしPADと類似のプログラム図式表現法について、これらは、前述のように、個別言語に依存しない手続き型言語の主要な三つの制御構造の図式化に主眼を置いている。そのため、PADエディタによるプログラムの入力は、個別言語に独立な制御構造部品の図形による入力と、各制御構造部品の中身のプログラムの文字による入力の、二段階にわかれている。図形部分の入力は、前記のように、画面上に表示した部品メニューから図形部品を選択することにより行なう。

【0012】PADのように基本的な図形部品が3個程度であれば、メニューに並ぶ部品はバリエーションを含めても10個程度なので、メニューに表示するのも、そこから選択するのも容易である。しかし、先願（特願平2-056360）または大島義光「LISPプログラムの図式表現VEX」（情報処理学会記号処理研究会資料54-3（1990. 3. 12））に記載したような、プログラミング言語の要素を全て図式により表現し

ている場合、図形部品の数100のオーダーになり、メニューに表示することは困難になる。また、表示出来たとしても、そこから選択することはさらに難しい。

【0013】また、上記のように、プログラムの入力、メニュー選択による図形部分の入力と、文字入力による詳細部分の二つにわかれており、マウスとキーボードを交互に使わなくてはならず、操作が煩わしい。キーボードまたはマウスのどちらかだけで、プログラムを入力することが出来れば、より効率的なプログラム作成が可能となるであろうことが予想される。

【0014】一方、前記したテキスト・ベースの構文主導型エディタ (Cornell Program Synthesizer) は、言語要素の構文テンプレートを使用することによって、正しい構文のプログラムの作成を支援することが出来るが、構文テンプレートの表現が文字だけによっているため (構文テンプレートは見かけは通常のテキストによるプログラムと同じ)、図式で表現したプログラムと比べて見にくい、という問題がある。また、プログラムの編集をこの構文テンプレートを基本として行なうため、編集操作に手間がかかるという問題がある。特に、既に作成したプログラム中のテンプレートの変更、別のテンプレートの挿入などの操作に手間がかかる。

【0015】先願 (特開平1-118925) に記載した構文主導型エディタは、この操作の手の問題点を改善しているが、改善点は主としてプログラムの入力に限定されている。エディタの各種編集操作—挿入、移動、複写、削除等—に対する構文情報の利用法については記載されていない。また、文字ベースのプログラムを対象にしている点で、見にくいという問題は上記Cornell Program Synthesizerと同様である。

【0016】前記図式ベースの構文主導型エディタは、構文主導型エディタと言いつつも、構文テンプレートのような言語要素の構文をガイドしてくれるような仕掛けはなく、入力したプログラムの構文を後からチェックしてくれる機能があるのみである。

【0017】本発明の第1の目的は、図形で表現したプログラムを、文字すなわちキーボードのみで、また特別な図形指定を行なうことなしに、入力する方法を提供することにある。

【0018】また、本発明の第2の目的は、図式で表現した構文テンプレートを用いることにより、見やすくかつ入力編集操作が容易で、同時に構文的な誤りの少ないプログラムを作成可能なプログラム・エディタを提供することにある。

【0019】

【課題を解決するための手段】上記目的を達成するために、本発明では以下の方法を提供する。

【0020】(1) 図形形式でプログラムを表示し入力編集を行なう図式プログラム入力編集方法に関し、図式プログラム表示画面上の任意の部分図式プログラム入力

可能位置に文字列入力可能な領域を設ける手段と、図式プログラム部品の名称もしくは図式プログラム部品を指示するキーワードと図式プログラム部品との対応表を具備し、該名称もしくは該キーワードを表わす文字列を該領域に入力後、特定のキーまたはボタンの押下を契機として、該名称または該キーワードに対応する図式プログラム部品を、該領域位置に、該領域と置き換えて表示する。

【0021】このとき、表示した図式プログラム部品が引数を持つならば、引数の名称 (仮引数名称) を示す文字列を、該図式プログラム部品の内部または近傍に表示する。

【0022】(2) 図形形式でプログラムを表示し入力編集を行なう図式プログラム入力編集方法に関し、名称またはキーワードを表わす文字列をその一部として持つ図式プログラム部品の、該文字列を変更し、部品変更を指示する特定のキーまたはボタンを押下したとき、該図式プログラム部品を変更後の名称またはキーワードに対応する図式プログラム部品に変更し表示する図式プログラム入力編集方法。

【0023】(3) 図形形式でプログラムを表示し入力編集を行なう図式プログラム入力編集方法に関し、図式形式プログラム表示画面上の任意の部分図式プログラム入力可能位置に、文字列入力可能な領域を設け、該領域に文字形式によるプログラムを入力後、特定のキーまたはボタンの押下を契機として、該文字プログラムを、対応する図式プログラムに変換し表示する図式プログラム入力方法。

【0024】(4) 上記(1)の図式プログラム入力方法において、新たな図式プログラム部品の入力用文字列入力領域の挿入コマンド発行後、図式表現プログラム上で該文字列入力領域を挿入可能な位置に図形カーソルを位置付けたとき、該挿入可能位置に対応する図式プログラムの部分を識別表示する図式プログラム入力編集方法。

【0025】このとき、文字列入力領域を挿入不可の位置に図形カーソルを位置付け、挿入実行を指示する特定のキーまたはボタンを押下したときは、警告音、警告メッセージ等の警告情報を発する。

【0026】また、挿入可能な位置で、挿入実行を指示する特定のキーまたはボタンを押下したときは、該挿入位置によって決定される図式プログラム部品を、該位置に挿入表示する。

【0027】(5) 図形形式でプログラムを表示し入力編集を行なうプログラム入力編集方法に関し、あらかじめ退避領域に退避している部分図式プログラムに対するペースト・コマンド発行後、もしくは図式プログラム上であらかじめ指定した部分図式プログラムに対する移動または複写コマンド発行後、図式プログラム上で該部分図式プログラムをペースト、移動または複写可能な位置

に図形カーソルを位置付けたとき、該ペースト、移動もしくは複写可能位置にある図式プログラムの部分を識別表示する図式プログラム入力編集方法。

【0028】このとき、ペースト、移動または複写不可の位置に図形カーソルを位置付け、ペースト、移動または複写実行を指示する特定のキーまたはボタンを押下したときは、警告音、警告メッセージ等の警告情報を発する。

【0029】(6) 図形形式でプログラムを表示し入力編集を行なうプログラム入力編集方法に関し、図式プログラム上であらかじめ指定した一つまたは複数の部分図式プログラムに対する削除コマンド発行により、該一つまたは複数の部分図式プログラムを図式プログラム上から削除後、残っている図式プログラムの各部を対応させつつ、詰合せを行ない、その結果として構文的に誤りのある部分図式プログラムが発生したとき、該部分図式プログラムを識別表示する図式プログラム入力編集方法。

【0030】

【作用】上記(1)(2)および(3)の方法により、キーボードのみを用いて、図式表現プログラムを作成することが可能となる。

【0031】また、(1)の方法により、言語要素の図式表現による構文テンプレートが表示されるので、誤りのないプログラムを容易に入力することが可能となる。

【0032】さらに、(4)(5)および(6)の方法により、図式表現プログラムの編集と並行してプログラムの構文チェックが行なわれるので、構文的な誤りのないプログラムを容易に作成することが可能となる。この方法はまた、構文情報の厳格な適用による編集操作の制限もないので、編集操作の効率を落とすこともない。

【0033】

【実施例】以下本発明を実施例により詳細に説明する。

【0034】以下の実施例中で用いるプログラムの図式表現には、先願(特願平2-056360)または大島義光「LISPプログラムの図式表現VEX」(情報処理学会記号処理研究会資料54-3(1990.3.12))に記載した表現を主として用いる。また、図式表現のベースとなるプログラミング言語としては、LISP言語(特にCommon LISP)を主に用いる。ただし、本発明は、LISP言語ならびにそれに基づく図式表現に限定されるものではない。また、文字表現プログラムの存在を前提とする図式表現プログラムに限定されるものでもなく、図式表現しか持たない、いわゆる視覚言語によるプログラムにも適用可能である。

【0035】図1に、本発明による図式プログラム入力編集方法を実現するシステムの構成を示す。

【0036】図1で、10は、パーソナル・コンピュータ、ワークステーション等の制御装置である。2は、図形を表示するためのディスプレイ装置である。3は、文字を入力するためのキーボード、4は、ディスプレイ画

面上の位置や、画面上に表示されている図形を何らかの編集対象として指定するための、マウス等の位置指示装置である。

【0037】図1の20は、本発明による図式プログラム入力編集プログラムである。本プログラムは、文字表現から図式表現への変換プログラム12、図式表現から文字表現への逆変換プログラム14、および入力編集制御プログラム15から構成されている。

【0038】図1の11は、ユーザが作成した文字表現によるソース・プログラムのファイルである。このファイル中のソース・プログラムを、文字表現から図式表現への変換プログラム12により、図式表現プログラム(のファイル)13に変換し、入力表示制御プログラム6を通じてディスプレイ装置8の画面に表示する。

【0039】図式表現から文字表現への逆変換プログラム14は、逆に図式表現プログラム13を文字表現によるソース・プログラム11に変換する。図式表現レベルでプログラムを入力編集後、この逆変換プログラムで文字表現によるプログラムを得れば、その後はこの文字表現プログラムに対するコンパイラで機械語に変換したり、またはこの文字表現プログラムに対するインタプリタを用いて、プログラムを実行することが出来る。もちろん、図式表現プログラムのコンパイラまたはインタプリタがあれば、文字表現に変換することなく、プログラムを実行することも可能である。

【0040】入力編集制御プログラム15は、キーボード3およびマウス4などの入力装置から入力された文字データ、ファンクション・キーのコード、コマンド、画面上の図形カーソル位置などの情報を入力表示制御プログラム6を通じて取り入れ、解釈し、また、図式プログラム部品表16を利用しつつ、入力編集処理を進め、適宜、プログラム12、14を起動する。

【0041】なお、図式プログラム部品表16には、逐次実行、条件判定、繰返しなどのプログラム構造の記述要素、あるいは関数などの、プログラムを構成する構成要素の文字表現とそれに対応する図式表現の両者を組にして登録してある。

【0042】また、入力表示制御プログラム6は、最近のパーソナルコンピュータやワークステーションにおけるウィンドウ・システム(例えばXウィンドウ)により実現することが出来る。

【0043】次に、具体的な図式表現プログラムの画面表示例、およびその上での入力編集操作方法について説明する。

【0044】(1) 入力例(その1) 関数remの場合(その1)

図2は、本発明による図式プログラム入力方法の一例を示した図である。同図(a)の50はプログラム入力画面、51は画面中に既に入力されているプログラムの一部である、1入力/1出力の関数fooを表わしている

図形（こゝでは箱）である。52はキーボードからの文字列（テキスト）入力可能な領域を示し、矢印58により図形51に連結されている。53は次に文字が入力される位置を表わす文字カーソルである。図では、文字列入力領域52に既に文字列remが入力されている。

【0045】文字列入力領域52は、図2では点線で囲った図形にて表示しているが、これに代えて、この領域の色を他の部分と変えて表示してもよい。また、領域の存在が明らかであれば、このような視覚的特徴を付けないでおいてもよい。これらのことは以下の説明でてくる入力領域に関しても同様である。

【0046】図1(a)の状態、次に、キーボード上に設けている変換キーまたはそれに代わるキーを押すと、同図(b)のような表示に変わる。(a)の文字列入力領域52が消えて、関数remに対する図形によるテンプレート、すなわち、図形54と、その二つの引数のための文字列入力領域55および56と、それらを図形54に連結する矢印55A、55Bが表示される。ちなみに、関数remは、numberをdivisorで割って、その余り(remainder)を返すCommon LISPの関数である。このように、本実施例では、関数remの引数を入力する領域が、自動的に、この関数remを表わす図形に対して、所定の相対的位置関係にある画面上の位置に表示される。とくに、これらの領域は、矢印55A、55Bでもって図形54に連結され、それにより、これらの領域が、図形54との密接な関係にあることを分かりやすくしているが、領域55、56は必ずしも図形54に連結させなくてもよい。

【0047】なお、領域55、56は、図2では点線で囲った図形にて表示しているが、これに代えて、この領域の色を他の部分と変えて表示してもよいことは、図2の領域52の場合と同じである。領域の存在が明らかであれば、このような視覚的特徴を付けないでおいてもよい。

【0048】文字列入力領域55および56内には、それぞれ仮引数名称number、divisorが表示されている。この仮引数名称numberおよびdivisorは、ユーザが入力した文字と区別するために、イタリック体(斜体)で表示される。この仮引数名称表示により、ユーザは、次に入力すべきものが何であるかを容易に判断することが出来る。また、次の入力を容易にするために、関数remの第1引数を表わす文字列入力領域55の先頭に、文字カーソルが移動している(図の57)。この状態で更にキーボードからの入力を進めると、文字列入力領域55内の仮引数名称numberは消去され、キーボードから入力された文字列に変わる。また、一旦入力した文字列を消去した場合、仮引数文字列が再び表示される。

【0049】文字列入力領域55内に次に表示したいプ

ログラムの図形部品の名称を入力し、再度変換キーを押すと、上記と同様、文字列入力領域55が消えて、その位置に入力した文字列により指定されるプログラムの図形部品が表示される。

【0050】(2)入力例(その2)関数remの場合(その2)

図3は、関数remに対する図式テンプレートの他の例である。本例では、関数remの仮引数名称numberとdivisorを、引数を入力するための文字列入力領域内でなく、関数を表わす箱54Aの中に、かつ、それぞれの引数を入力するための文字列入力領域55、56と対応させて示している。

【0051】この仮引数名称は、通常、入力するプログラム部品がシステムであらかじめ用意している言語要素、すなわち組み込み関数等(この例では関数rem)であるとき、表示される。この他、プログラム言語の処理系がインタプリタ形式のものであって、指定した要素に対するプログラム定義がメモリ中にあらかじめ読み込まれている場合にも、その情報を参照することにより、仮引数名称を表示することが出来る。

【0052】(3)入力例(その3)プログラム入力の場合

上記の例では、文字列入力領域52にプログラム言語の要素の名称(この例では“rem”)を入力して変換キーを押した。その代りに文字形式の完全なプログラムを入力することもできる。

【0053】例を図4に示す。

【0054】同図(a)は、図2(a)の文字列入力領域52中に、プログラム“(* x (+ y z))”

(これはLISP言語で式“ $x * (y + z)$ ”を実行するプログラムを表わす)を入力した状況を示し、次に変換キーを押すと、同図(b)のような図式によるプログラムに変換される。すなわち論理積、論理和を表わす箱63、64と、引数x、y、zと、これらに関連づける矢印とを有する図形を表示する。

【0055】入力するのは完全なプログラムでなくてもよい。引数に不足がある場合、不足している引数に対応する文字列入力領域をシステムが自動的に補う。また、余分の引数が指定されている場合には、図式に変換したプログラム上でその引数をシステムが強調表示し、ユーザに注意を促す。構文の正しくない引数を入力して変換したときにも、同様にその引数に対する図式表現をシステムが強調表示して、ユーザに注意を促す(あるいはまた、図式表現に変換出来ない場合には、文字のまま表示し、そこを強調表示する)。例を図11に示す。

【0056】図5(a)は、図2(a)の文字列入力領域52内に、文字列remの代わりに、文字列“(rem (elt list) 3 8)”を入力した状態を示している。この状態で変換キーを押すと、同図(b)のような図式表現プログラムに変わる。関数remは本

来2引数の関数であるので、余分な引数“8”がハッチングされて強調表示されている(図5(b)の61)。また、関数eltも2引数の関数であるが、1引数しか与えられていないので、残りの引数名称indexを含む文字列入力領域62が補われて表示されている。ちなみに、関数eltは、リストまたはベクトルによる形式で与えられたデータから、index(数)で指定した要素を取り出すCommon LISPの関数である。

【0057】なお、本図の場合のように、余分な引数(図5(b)の61)が残っていたり、必須引数(図5(b)の62)に入力が与えられない状態のまま編集を終了しようとすると、警告音、警告メッセージ等でユーザに注意を促す。

【0058】(4)入力例(その4)関数letの場合図6は、本発明による図式プログラム入力方法の他の例を示した図である。本図は、図2(a)で、文字列入力領域52にremの代わりにletと入力し、変換キーを押した後の状態を示している。

【0059】同図70の枠とそれに囲まれた部分は、Common LISPのブロック構造を表わす言語要素letを図式により表現したものである。71はlet(の図式表現)であることを示すタイトル部分、72はブロック構造letにより導入される局所変数の名称を入力する文字列入力領域、73はその初期値を示すプログラムを入力する文字列入力領域、74はブロック構造letによるプログラム本体を入力する文字列入力領域を、それぞれ示している。文字列入力領域72、73、74内には、それぞれの仮引数名称を示す文字列var、init、formが、イタリック体で示されている(varはvariable(変数)、initはinitial value(初期値)のそれぞれの略称、formは、実行可能な部分プログラムを表わすCommon LISP特有の名称である)。これらのイタリック体で表示された文字列は、前記と同様それぞれの文字列入力領域内にユーザが文字を入力すると、消去される。また、一旦入力した文字列を消去した場合、仮引数文字列が再び表示される。

【0060】図2(b)と同じく、次の入力を容易にするために、局所変数名称の文字列入力領域72の先頭に、文字カーソル76が移動している。

【0061】図6の縦棒75は逐次実行による処理を表わす図式表現である。75の縦棒を縦に伸ばし、formによる部分プログラムを複数並べてこの縦棒につなぐことにより、複数のformによる処理を上から順に実行するプログラムを表現することが出来る。

【0062】ブロック構造letでは、一般に複数の局所変数を設定することが出来る。

【0063】図7(a)は、図6の状態、varに変数名xを入力して変換キーを押し、次に文字列入力領域73に初期値1を入力した状態を示している。この状態

で変換キーを押すと、同図(b)のように、局所変数xの下に、新しい局所変数を入力する文字列入力領域77と、初期値を入力する文字列入力領域78が追加される。

【0064】局所変数は任意個設定可能であるから、上記の操作を繰り返すことによって、次々と新しい変数と初期値の文字列入力領域を追加することが出来る。

【0065】新しく追加された局所変数と初期値の文字列入力領域にともに何も入力しない状態で単に変換キーを押せば、追加表示された局所変数と初期値の文字列入力領域が消去され、新しい局所変数の追加を終了することが出来る。

【0066】なお、追加表示されているが入力がなされていない局所変数の文字列入力領域をそのままにしておいて、他の作業を行なうことも可能である。この場合は、局所変数および初期値の文字列入力領域はそのまま残る。再度この文字列入力領域内に文字カーソルを移し、何も入力しないまま変換キーを押せば、上記と同様、追加表示されている局所変数と初期値の文字列入力領域が消去される。また、消去せずに新しい変数名を入力することもできる。

【0067】追加表示されている局所変数(および初期値)文字列入力領域に対して特に何もせず、編集終了の操作をしたときは、この追加領域は存在しないものとして取り扱われ、プログラムはその状態でファイル等に格納される。

【0068】Common LISPでは、このような複数の局所変数を設定可能な言語要素は、このほか、関数定期の言語要素defun、繰り返しの言語要素do、などがある。それぞれ、複数の仮引数、および複数の繰り返しの制御変数を設定することが出来る。

【0069】(5)入力例(その5)条件文ifの場合図8は、本発明による図式プログラム入力方法の他の例を示した図である。同図(a)は、図2(a)で、文字列入力領域32にremの代わりにifと入力し、変換キーを押した後の状態を示している。

【0070】図8(a)で、内部にifと書かれた枠(図の80)とそれから右の図形部分から構成される複合図形は、Common Lispの条件分岐を表わす言語要素ifを図式により表現したものである。81は、配管等の設計図でよく用いられるバルブを表わす図形を流用したもので、条件分岐ifの機能を象徴的に表わした図形である。82は、分岐のテスト条件を表わすプログラムを入力するための文字列入力領域、83は、テスト条件が成立するときに実行されるプログラムを入力する文字列入力領域、84は、逆にテスト条件が成立しなかったときに実行されるプログラムを入力する文字列入力領域である。前記と同様、それぞれの仮引数名称(test、then、else)が、文字列入力領域内にイタリック体で表示されている。

【0071】図の引数 `else` は付加的な引数で、入力してもよいし、しなくてもよい。条件を表わすプログラム “(> x 1)” (これはLISP言語で式 “ $x > 1$ ” を実行するプログラムを表わす) を、`test`、すなわち文字列入力領域82に入力して変換キーを押し、次に “(+ x y)” (これはLISP言語で式 “ $x + y$ ” を実行するプログラムを表わす) を、`then`、すなわち文字列入力領域83に入力して、変換キーを押した直後では同図(b)に示す。図のように、比較を表す図形92、加算を表わす図93と、入力された引数 `x`、`1`、`x`、`y` がこれらの図形に矢印でもって結合した形で表示される。このとき文字カーソル85は、`else`、すなわち文字列入力領域84の中に来ている。この状態で変換キーを押すと、`else` の文字列入力領域54は消去され、プログラムの図式表現は、同図(c)のようになる。

【0072】この `else` に対する文字列入力領域84は、同図(b)の状態で別の編集に移った場合はそのまま残される。`else` の文字列入力領域84に対して特に何もせず、編集終了の操作をしたときは、前記局所変数の追加入力用文字列入力領域と同様に、この文字列入力領域はないものとして処理される。

【0073】なお、Common LISPでは、通常の間数においても、前記 `if` の引数 `else` と同じく、省略可能な引数を設定することが出来る。また、前記局所変数のリストと同様に、任意個の要素を持つ引数を定義することも、同じく可能である。

【0074】(6) 修正

次に、作成した図式プログラムを部分的に修正する場合の例について説明する。

【0075】図9および図10は、Common LISPの条件分岐の言語要素 `if` を、他の条件分岐の言語要素 `cond` に変更する例を示したものである。`if` は、前記のように、一つのテスト条件によって二つの実行部の一つを選択して実行するための機構である。図9は、図8の `else` 部84に対して、`x-y` を入力した後得られる図形94を有する。これに対し、`cond` は、複数の条件部と実行部のペアを記述し、実行時はこのペアを先頭から走査していき、条件部が成立した部分の実行部を実行する。

【0076】図9(a)の状態で、マウス等を使用してキーワード `if` の部分に文字カーソル90を置き、`if` を `cond` に書き換え(同図(b)91)、変換キーを押すと、図式表現プログラムは図10のようになる。`if` のときにあったテスト条件の部分(図9(a)92)は、図10の `cond` の条件部/実行部ペアの最初の部分に置かれる。また、2番目の条件部/実行部ペアの条件部は何も指定されていないので、新しい文字列入力領域95が追加され表示される。`if` のテスト条件が成立したときに実行される部分(`then` 部、同図93)と

成立しなかった場合に実行される部分(`else` 部、同図94)は、それぞれ二つの条件部/実行部ペアに分けて置かれている。

【0077】この例では、変更前と変更後の言語要素は似たような機能を果たすものであったが、一般に変更後の言語要素は変更前の言語要素と特に関係がある必要はなく、どのようなものでもよい。ただし、変更前の言語要素の引数と変更後の言語要素の引数とは一般には対応関係がないので、システム側で適当に対応を付けることになる。その場合、上記でも一部発生しているが、引数の過不足が生じる可能性がある。これに対して、前記した誤りを含む文字プログラムの図式プログラムへの変換と同様に、不足している引数は補い、過剰な引数は強調表示してユーザに注意を促す。

【0078】プログラムのもう少し大きな修正を行なう場合は、通常後述の図式ベースの編集操作により行なう。しかし、次のような方法も可能である。

【0079】まず、修正したい領域を一括して文字形式のプログラムに逆変換する。次に、文字ベースで修正を行なった後、再度図式ベースの表現に戻す。具体的には、図4(b)のようなプログラムで、関数* (図4(b)の63)をマウス等で指定し、図式表現から文字表現への変換(逆変換)のコマンドを実行すると、図4(a)のような表示が得られる。この状態で、文字ベースでプログラムの編集を行ない、修正後変換キー等によりもとの図式表現プログラムに戻す。

【0080】(7) 挿入

次に、図式プログラムの編集操作の例について説明する。

【0081】図11は、新しい文字列入力領域の挿入操作の例を示した図である。

【0082】ところで、通常、テキスト・ベースのエディタでは、任意の位置に文字列を挿入してプログラムを修正することが出来る。しかし、本発明のような図式表現によるプログラム・エディタでは、途中に新しいプログラムを挿入するためには、挿入する位置に入力を行なうための領域、すなわちこれまでの説明で用いているような文字列入力領域を明示的に設ける必要がある。ところが、本発明の図式プログラム・エディタでは、図式による個々のプログラム部品は、対応する言語要素のテンプレート、すなわち構文形式を表現しているので、まったく勝手な位置に挿入用の文字列入力領域を追加することは出来ない。そこで、この挿入用の文字列入力領域の設定にあたって、どこが挿入可能であるか、あらかじめユーザに判るようになっていなければならない。

【0083】図11は、このための支援機能を説明した図である。

【0084】図11(a)は、エディタ・ウィンドウに付属するプルダウン・メニュー(図示せず)等で挿入コマンドを指定し、次にマウス等の画面上の位置を指示す

る装置を用いて、図形カーソル102を挿入したい位置に移動した状況を示している。この例では、挿入したい位置は、二つの関数 `fun1` (図の100) と `fun2` (図の101) の間である。この状況で、マウス・ボタン等を押して挿入コマンドの実行を指示すると、図11(b)のように、挿入用の文字列入力領域104が図形カーソル102で指定した位置に挿入され、その内部に文字カーソル53が表示される。

【0085】ところで、図11(a)では、`fun2` から `fun1` を結ぶ矢印103が太線になっている。このように、本実施例では、図形カーソルが位置の矢印103の上におくと、これをたく表示するようになっていく。この状況で挿入操作を実行しないまま(マウス・ボタン等を押さず)マウス等操作して図形カーソルを矢印103の上から離すと、矢印103は通常の太さの表示に戻る。

【0086】このように、挿入コマンドを指定した後、図形カーソルを挿入用文字列入力領域の挿入可能な位置に移動したとき、そこが強調表示されるようになっていくと、ユーザはそこが挿入可能であることがわかり、編集操作が容易になる。

【0087】この例では、挿入可能な位置を示す図形の強調表示を、矢印の線を太くすることによって行なっているが、これは他の方法によって行なってもよい。例えば、矢印の線を点線または破線で表わす、線の色を変える、などの方法がある。

【0088】さらに、本実施例では、上記の挿入コマンド実行状態にあるとき、同図(a)に示しているように、図形カーソル102は挿入コマンド実行状態を示す形として、矢印の後ろに小さな四角形を付けた形になる。通常は、単なる矢印などの形をしている。

【0089】図12は、挿入用文字列入力領域の挿入操作の別の例である。関数 `fun1` (図12(a)の110) の二つの引数(関数 `fun2` (111) と関数 `fun3` (112)) の間に別の引数を挿入しようとした例である。挿入用の図形カーソル113を関数 `fun1` を示す箱の、二つの引数を示す矢印の先端によってはさまれている辺の一部114の上に置くとこの一部が太線になって表示される。

【0090】この状況で、マウス・ボタン等を押すことによって、挿入コマンドを実行すると、図12(b)に示すように、新しい引数を入力する文字列入力領域115が挿入され表示される。カーソル53はその内部に表示される。

【0091】図12(a)による例では、関数 `fun1` の最初の引数(関数 `fun2`)の前、または最後の引数(関数 `fun3`)の後に、文字列入力領域を挿入することも可能である。最初の引数の前に挿入するときは、関数 `fun1` の箱の左辺上端から第1引数の矢印の先端までの間が強調表示され、最後の引数の後に追加するとき

は、関数 `fun1` の箱の左辺下端から最後の引数の矢印の先端までの間が強調表示される。

【0092】図13に、挿入用文字列入力領域の挿入操作の第3の例を示す。Common LISPのブロック構造を表わす言語要素 `let` 120の局所変数リストの途中に別の局所変数を新しく追加する場合の例である。

【0093】図13(a)では、挿入用の図形カーソル121が、変数リストに既に設定されている二つの変数 `var1` (図の122) と `var2` (図の123) の間を指している。このとき、二つの変数を仕切っている線が太い実線で表示される(図の124)。

【0094】この状態で、マウス・ボタン等を押して挿入コマンドを実行すると、図13(b)のように、局所変数名を入力する文字列入力領域125と、その初期値を表わすプログラムを入力する文字列入力領域126が挿入され表示される。

【0095】この例でも、図12の場合と同様に、変数リストの先頭または最後に新しい局所変数用の文字列入力領域を挿入することも可能である。このときは、変数リストの上端または下端が強調表示される。

【0096】図14は、挿入用文字列入力領域の挿入操作を示す第4の例である。Common LISPの条件分岐の言語要素 `cond` (図の130) の複数の条件部/実行部140、141のペアの途中に、新しい条件部/実行部ペアを追加する場合の例である。

【0097】図14(a)では、挿入用の図形カーソル131が、条件要素 `cond` のタイトルを表わしている箱の左端の線分上で、二つの条件部/実行部140、141のペアから発している矢印(図の132、133)の先端にはさまれる部分を指している。このとき、その線分の一部が太線になる(図の134)。

【0098】この状態で、マウス・ボタン等を押して挿入コマンドを実行すると、図14(b)のように、新しい条件部を入力するための文字列入力領域135と、新しい実行部を入力するための文字列入力領域136、および条件判定を示す図形137(`if`と同様、配管図のバルブ記号の流用)、などからなる新しい条件部/実行部ペアの図形が挿入され表示される。

【0099】前例と同様、この場合も、最初の条件部/実行部ペアの前、または最後の条件部/実行部ペアの後ろにも挿入可能であり、また、その際に対応位置の強調表示が行なわれることは言うまでもない。

【0100】以上の説明から明かなように、挿入操作の各例において、挿入される文字列入力領域は、それぞれ挿入される位置における構文に応じたものになっている。詳しく言えば、挿入される文字列入力領域は、挿入される位置の構文に応じて、一つまたは複数の文字列入力領域とその位置の構文に付随する図形から構成されている。これは、最初に述べた図形による構文テンプレートを用いたプログラム入力方法の、挿入操作における変

形と見ることもできる。

【0101】このように、挿入時には挿入場所に応じた適切な文字列入力領域が挿入されるので、挿入操作を容易に行なうことが出来る。同時に、構文的に正しいプログラムの作成を行なうことが出来る。

【0102】さて、図11～図14の例では、挿入可能位置を強調表示する例を示した。これに対し、ユーザが、このような挿入可能位置以外の場所で、挿入操作を実行しようとすることもありえる。

【0103】例えば、図12の例で、関数 `fun1` は2引数の関数で、それ以外に省略可能な引数または任意個の要素を持つことの出来る引数を持たないものとする。ちなみに引数の個数の情報は、プログラム部品がシステムで用意している言語要素、すなわち組み込み関数等であるときは、あらかじめわかる。また、ユーザ定義の関数等であっても、プログラム言語の処理系がインタプリタ形式のものであって、そのプログラム定義がメモリ中にあらかじめ読み込まれている場合にも、わかる。このような場合は、図12のようにさらに引数を追加することは出来ない。そのような場合には、図12(a)に示した位置に図形カーソルを位置付けても、図12(a)のように114の部分太線が表示されることはない。また、その状態でマウス・ボタン等を押して挿入コマンドを実行しようとしても、挿入処理は実行されない。その代りに警告音が発せられ、画面上の適切な位置に、その位置には挿入出来ない由の警告メッセージが表示される。

【0104】この例以外にも、挿入可能でない任意の位置で、ユーザが挿入コマンドを実行しようすると、同様に警告音と警告メッセージが発せられる。

【0105】以上で文字列入力領域の挿入操作の説明を終える。

【0106】なお、これまでに説明した図式プログラムの入力操作、ないしは文字列入力領域の挿入操作等で導入した文字列入力領域のうち、関数等の必須引数（必ず指定しなければならない引数）に相当するものがあり、それらに何ら入力を与えない状態で編集を終了しようすると、その必須引数に対応する文字列入力領域を強調表示するとともに、警告音および警告メッセージが発せられる。

【0107】以上の各例は、キーボードからの文字列入力を利用して、図式によるプログラムを作成編集する例であった。

【0108】これから先に述べる例は、キーボードからの文字列入力とは独立の、図式によるプログラム・エディタ一般に適用可能な例である。

【0109】(8) ペースト、移動、複写
まず、ペースト・コマンド（貼り付けコマンド）および移動・複写コマンドの操作について説明する。

【0110】ペースト・コマンドは、カット・コマンド

（切り出しコマンド）等により、エディタ内部の退避領域にあらかじめ退避している部分的な図式表現プログラムを、画面上に表示されている図式表現プログラムの一部に挿入するコマンドである。また、移動・複写コマンドは、画面上に表示されている図式表現プログラムの一部を別の場所に移動または複写するコマンドである。いずれのコマンドも、ある対象（部分プログラム）を、指定位置に移す点が共通している。

【0111】このペーストもしくは移動・複写コマンドには実は2種類の実行形式がある。一つは、前記文字列入力領域の挿入と同様に、ペースト・移動・複写対象を指定位置に挿入する実行形式、他の一つはペースト・移動・複写対象を他の別の対象と置き換える形でペースト・移動・複写する実行形式である。

【0112】挿入による実行形式では、エディタ・ウィンドウのプルダウン・メニュー等でペースト・移動・複写コマンドの一つを指定した後、対象を挿入したい場所に移動すると（なお、移動または複写コマンドの場合は、その前に移動・複写する対象を指定するステップがある）、その場所が挿入可能な場所ならば、前記文字列入力領域の挿入と同様に、図形カーソルの下にあるプログラムの図形部品の一部が強調表示される。その状態でマウス・ボタン等を押してコマンドの実行を指示すると、指定したコマンドの処理が実行される。また、図形カーソルのある場所が、対象の挿入可能でない場所のときは、その場所に図形カーソルを置いた状態でマウス・ボタン等を押してコマンドの実行を指示しても、コマンドは実行されない。その代り、警告音および警告メッセージが発せられる（ただし、以下の置き換え形式によりペースト・移動・複写コマンドを実行可能な位置に図形カーソルがある場合を除く）。

【0113】ここで、ペースト・移動・複写コマンドにより対象を指定位置に挿入可能か否かの判定は、前記文字列入力領域の挿入の場合と異なり、挿入場所にある図形部品の種別だけでは決まらない。挿入する対象の種別にも関係する。すなわち、図11(a)や図12(a)のような状況では、挿入可能な対象は、戻り値を返す一つの関数相当のもの、またはトップレベルがその関数と同等な複合プログラムである。図13(a)の場合には、局所変数とその初期値からなるペア、図14(a)の場合には、`Common LISP`の言語要素 `cond` を構成する条件部／実行部ペアである。このように、強調表示されるか否かは、図形カーソルが指している図形位置と挿入対象によって決定される。

【0114】次に、ペースト・移動・複写コマンドを、置き換え形式により実行する例について説明する。

【0115】その例を図15に示す。

【0116】図15(a)は、ウィンドウのプルダウン・メニュー等によりペースト・コマンドを指定後、図形カーソル140を置き換え形式でペーストしたい位置に

移動した状況を示している。図では、退避している対象をペースト・コマンドにより置き換えたい位置は（というよりは置き換えたい画面上の対象は）、関数 `fun1` の図形 100 の第 2 引数位置にある文字列入力領域 141 である。また、ペースト・コマンド実行前にカット・コマンド（切り出しコマンド）等でエディタ内部の退避領域に退避しているプログラムは、図 15（b）に示したものであるとする。この状況で、マウス・ボタン等を押してペースト・コマンドを実行すると、図 15（c）のように、図 15（a）の文字列入力領域 141 が、図 15（b）のプログラム図形 111 で置き換えられて表示される。

【0117】ところで、図 15（a）では、図形カーソルの下にある図形、すなわち図形 141 がハッチングをかけられて強調表示されている。これは、前記文字列入力領域の挿入の場合と同様に、図形 141 が、退避領域に退避されているプログラムと、ペースト・コマンドにより置き換え可能であることを示している。

【0118】この状況でペースト・コマンドを実行しないまま（マウス・ボタン等を押さず）マウス等を操作して図形カーソルを図形 141 の上から離すと、図形 141 は通常表示に戻る。

【0119】このように、ペースト・コマンドを指定した後、図形カーソルをペースト可能な対象の上に移動したとき、前記文字列入力領域の挿入の場合と同様、そこが強調表示されるようになっていいると、ユーザはそこが置き換え形式によりペースト可能であることがわかり、編集操作が容易になる。

【0120】図 15 の例では、画面上に表示されている置き換え対象は文字列入力領域であった。しかし、置き換え対象はこれに限られるわけではない。画面上に表示されている任意の対象をペースト・コマンドによって置き換えることが出来る。ただし、画面上に表示されている勝手な対象を、別のまったく勝手な対象で置き換えられるということではなく、両者は類似のものである必要がある。

【0121】ペースト・コマンドにより画面上の指定対象を置き換え可能か否かは、挿入形式によるペースト・コマンドの実行と同様、画面上の置き換え対象の種別と、退避領域に退避している対象の種別の両者によって決定される。すなわち、図 15（a）のような場合は、戻り値を返す関数相当のもの、またはトップレベルがその関数と同等な複合プログラムである。また、図 13（b）の 125 で示される局所変数が置き換え対象として指定されている場合には、置き換え可能なものは、局所変数ないし局所変数とその初期値からなるペアの形をした部分プログラムである。図 14（b）の `cond` の一つの条件分岐を表わす図形 136（`cond` の条件節の一つ）が置き換え対象として指定されている場合には、置き換え可能なものは、それと同じ形をした条件節

となる。このように、置き換え可能か否かは、図形カーソルが指している図形、すなわち被置き換え対象の種別と、エディタの退避領域内にあるプログラム、すなわち置き換え対象の種別の両方によって決定される。これによって、画面上に表示されている被置き換え対象を強調表示するか否かが決まる。

【0122】この例では、ペースト可能な対象を示す図形の強調表示を、ハッチングをかけることによって行なっているが、これは他の方法によって行なってもよい。例えば、図形を白黒反転表示する、あるいはカラー表示可能な表示装置を用いている場合には、図形に色を付けて表示する、などの方法を用いることが出来る。

【0123】なお、上記のペースト・コマンド実行状態にあるとき、図 15（a）に示したように、図形カーソルはペースト・コマンド実行状態を示す形になっている（通常の矢印に黒塗の小さい四角形が付いている）。

【0124】以上の説明では、もっぱらペースト・コマンドの処理について説明したが、移動または複写コマンドの場合もまったく同様である。すなわち、ペースト・コマンドで退避領域のある置き換え対象が、画面上の別の場所にある対象に変わるだけである。

【0125】これまでの説明で、ペースト・移動・複写コマンドの実行形式に挿入形式と置き換え形式があるとして、それぞれの処理について説明してきたが、それぞれの実行形式をユーザが意識する必要はない。図形カーソルがある位置によって、すなわち図形カーソルが挿入形式でコマンドを実行すべき図形の一部を指しているのか、または置き換え形式でコマンド実行すべき図形対象を指しているのかによって、どちらの実行形式かが決まる。ただし、どちらの場合にしろ、図形カーソルは何かの図形の一部を指すことになり、どちらの実行形式か判定するには曖昧である。しかし、これは、図形カーソルの位置によるそれぞれの実行形式の検出範囲を適切に設定することによって、区別することが可能である。また、置き換え形式による図形の検出は図形の全体、挿入形式による図形の検出は図形の一部であるから、どちらか曖昧なときは、挿入形式による図形の検出を優先させればよい。システムがどちらの実行形式として認識しているかは、図形カーソルがある位置の図形全体が強調表示されるか、または挿入可能な位置を示す図形の一部が強調表示されるかによって示されるので、ユーザは、実行形式の区別を容易に判断することが出来る。しかし、挿入形式と置き換え形式のそれぞれ強調表示を、色などによって区別することにすれば、さらにわかりやすくなることが出来る。

【0126】なお、以上の挿入、およびペースト、移動、複写コマンドの各例において、図式プログラム部品のマウス・カーソルの下にある部分だけが強調表示されるとして説明した。しかし、これを、マウス・カーソル下にある図式プログラム部品の、コマンド適用可能な部

分を全て強調表示するようにすることもできる。この方がユーザに対してより親切かもしれない。さらに、挿入、ペースト、移動、または複写コマンドが起動されたとき、画面上に表示されている図式プログラム中の全ての図式プログラム部品のうち、コマンド適用可能な部分を全て強調表示するようにすることもできる。この場合は、コマンド適用可能な部分が一目で見渡せるので便利である。

【0127】(9) 削除

次に、図式表現プログラムの部分的な削除を行なうコマンドについて説明する。

【0128】図16にその例を示す。

【0129】図16(a)は削除コマンド実行前の状況を示した図である。図の151(関数fun2)が削除対象である。マウス等であらかじめ選択操作を施し、その結果、識別表示されている。ここで削除コマンドを実行すると、同図(b)のような表示になる。関数fun2が削除された結果、関数fun1(図の150)と、関数fun2の引数であった関数fun3(図の152)および関数fun4(図の153)の間が詰められて、関数fun3と関数fun4は、関数fun1のそれぞれ第2引数および第3引数として接続されている。

【0130】ここで関数fun1は2引数の関数であるとする。その場合、関数fun3は関数fun1の第2引数となっているので、構文的に問題ない。しかし、関数fun4は関数fun1の第3引数位置に来ているので、関数fun1の余分な引数となっている。この状況を示すために、図16(b)では関数fun4を強調表示し(図の153)、ユーザに対し、関数fun4は構文的に誤りとなる位置にあることを示している。

【0131】このように、削除コマンドの実行結果、構文的に誤りとなる部分プログラムが発生した場合、上記の処理によって、ユーザに対して構文的に誤っている部分の所在を示すことができる。これによって、ユーザは構文的な誤りのないプログラムを作成することが出来る。

【0132】一方、削除コマンドの実行により削除対象が削除され、その上位の図形部品につながるものが何もなくなくなってしまうときがある。このときは、その位置に未入力状態の文字列入力領域を接続する。

【0133】ところで、以上に述べた図式プログラムの入力編集方法は、上記の各種実施例で用いてきた図式表現法によるプログラムだけでなく、他の方法による図式表現プログラムにも適用可能である。

【0134】例えば、ベース言語にC言語を用い、図式表現法として従来例で引用したPADを利用する場合を考える。

【0135】図17に一例を示す。

【0136】同図(a)の160は文字列入力領域である。今この文字列入力領域160に文字列“if”が入

力されている。この状態で変換キーを押すと、同図

(b)のようになる。同図(b)の161は、条件分岐を表わすPAD図形である。また、162は、分岐条件の内容を記すための文字列入力領域、163、164はそれぞれ条件が成立したとき(YES)、および成立しなかったとき(NO)に実行されるプログラムを入力する文字列入力領域である。次の入力を促進するため、文字カーソルが文字列入力領域162の先頭に移動している。

【0137】図18に別の例を示す。

【0138】同図(a)は文字列入力領域170に文字列“switch”と入力した状態を示しており、この状態で変換キーを押すと、同図(b)のようになる。

【0139】C言語のswitch文は、式の値にしたがって、後続するいくつかの文(任意個)の一つに制御を移すためのものである。後続の各文は、case接頭辞により式の値と比較するための定数値を持つ。switch文の式の値とcase接頭辞の定数値が一致したとき、対応する文の本体が実行される。

【0140】図18(b)の171は、図17(b)の161と同じく、条件分岐を表わすPAD図形である。172はswitchに付属する式を入力する文字列入力領域、173、175は、それぞれ文の条件照合用定数値を入力する文字列入力領域、174、176は、式の値が173、175の定数値と一致したとき実行されるプログラムを入力する文字列入力領域である。

【0141】図18(b)の状態ですべての文字列入力領域に必要な内容を入力し(同図(c)の状態になる)、変換キーを押すと、同図(d)のように、新しいcase定数値と対応する実行プログラムの文字列入力領域が追加される(同図177、178)。これは、図6で説明した例と同様な機能を利用したものである。同じ操作を繰り返すことにより、このcase定数値と実行プログラムの文字列入力領域のペアは、好きなだけ追加することができる。

【0142】また、新しく追加されたcase定数値と実行プログラムの文字列入力領域の両方共に何も入力しない状態で単に変換キーを押すと、追加表示されたこのペアは消去され、このペアの新しい追加を終わることが出来る。

【0143】このほか、図4および図5で説明したような、文字列入力領域にプログラムを入力して、これを図式表現に変換する方法は、PADの場合にも容易に適用可能である。

【0144】以上、本発明による図式プログラム入力編集方法のさまざまな例について説明した。

【0145】(10) 入力編集プログラム

次に、これら図式プログラムの入力編集方法を具体的に実現する方法について述べる。

【0146】図19は、図1の入力編集制御プログラム

15の基本制御フローを示すPAD図である。図19は、図式表現プログラムのエディタを起動し、画面にエディタ・ウィンドウが表示されている状況にあるときのフローを示している。

【0147】図19からわかるように、最初は、キーボード、マウス等の入力装置からのユーザ入力待ちにあり、入力が入ると対応する処理を行ない、また入力待ちに戻るというループをなしている。

【0148】ステップ200で、キーボードまたはマウス等から入力が入ると、ステップ201に進み、入力の種別の判定を行なう。入力がキーボードからの入力ならば、ステップ202に進み、キーボードからの入力処理を行なう。マウス・ボタン押下による入力ならば、ステップ203に進む。また、エディタ・ウィンドウのプルダウン・メニューに用意されているコマンドが選択されたことを示す入力ならば、ステップ204に進み、コマンドの処理を行なう。

【0149】図20は、キーボード入力処理を示すフローである。

【0150】まず、ステップ210で、キーボードからの入力の判別を行なう。入力が文字データならばステップ211に進み、文字入力処理を行なう。入力が変換キーならばステップ212に進み、文字列入力領域に入力されているキーワードを対応する図式テンプレートに変換する。入力がキーボード上の各種編集キー（例：挿入、削除、後退など）ならばステップ213に進み、入力された編集キーに対応する処理を行なう。

【0151】次に、文字入力および変換処理について、図2の例を参照して説明する。

【0152】図21は、図1の図式表現プログラム・ファイル13の内容を示したものである。ここでは、例として、図2(a)に表示されている図式プログラムに対応する内部データ527を示している。

【0153】図21の300は、図2(a)の関数f o oの図形51に対する内部データである。以下このような画面上の各種図式プログラム部品に対する内部データを、「セル」と呼ぶ。セル300は、関数f o o 51を画面に表示するために必要なデータを格納する複数のスロットを持っている。

【0154】最初のスロット「分類」には、セルの種別を表わす名称が入る。この場合は関数であるから、関数を指示する名称「関数」が入っている。

【0155】2番目、3番目のスロット「位置」「サイズ」には、関数f o oの図形を画面上に表示するときの位置およびサイズの情報の情報、具体的には2次元の座標値および寸法が入っている。

【0156】4番目以降のスロットは、関数を表現する図形部品固有のスロットである。

【0157】スロット「名称」には関数の名称を表わすデータを保持するセル301へのポインタ、スロット

「引数」には関数の引数を表わすデータを表わすセル302をポイントするポインタ305へのポインタ、スロット「枠」には関数の具体的形状を表わしている枠の図形データを表わすセル308へのポインタ、スロット

「戻り値矢印」には関数の戻り値を表わす矢印の図形を表わすデータを表わすセル303へのポインタが、それぞれ入っている。

【0158】例えば、スロット「名称」には、文字列“f o o”そのものではなく、文字列を表現するデータを保持するセル301を指すポインタが入っている。部品名称を表わすデータを独立のセルにしているのは、関数の名称を表わす文字列が「関数」のセルとは別に独自の「位置」「サイズ」の情報を持つこと、また、関数の名称は固定的なデータでなく、これもまた入力編集の対象になること、が理由である。

【0159】セル301は、固有のスロットとして「文字列」等のスロットを持っている。

【0160】セル301の「分類」の内容は、単なる「文字列」ではなく、「文字列入力領域」となっている。これは、上記のように、関数の名称（この場合“f o o”）をあとから修正変更できるようにしているためである。「文字列入力領域」に関するさらに詳しい説明は後述する。

【0161】セル300のスロット「引数」には、セル305を介してセル302が接続されている。セル302は、図2(a)の文字列入力領域52に対応するセルである。スロット引数にセル302を直接接続していないのは、関数の引数は一般に複数個存在するためである。セル305は、それら複数の引数をつなげて一つにまとめるための補助的なデータである。

【0162】セル300のスロット「枠」には、セル308が接続されている。セル308は、長方形を表示するためのデータで、この場合、図2(a)の関数f o o(51)の図形表現である関数の箱を表示するために用いられている。

【0163】セル308には、「分類」「位置」「サイズ」の各スロットに加えて、「枠の形状」「枠の太さ」「枠の色」の各スロットがある。それぞれ長方形の枠を表わしている線の形状—実線、点線、破線等の種別—、太さ、色の情報が格納される。

【0164】セル300のスロット「戻り値矢印」には、セル303の矢印のセルが入っている。このセル303は、「始点」「終点」「形状」「太さ」「色」の各スロットを持つ。これらのスロットは、図4に示した線分のセルが持つスロットと同じである。矢印は、その基本部分は図形的に線分と共通で、やじりの飾りの部分がこれに加わった形になっている。したがって、矢印は、線分と共通のスロットで表わすことができる。

【0165】セル302は、固有のスロットとして、「文字列」「戻り値矢印」の各スロットを持つ。

【0166】セル302のスロット「文字列」には、キーボードから入力された文字列が入る。スロット「戻り値矢印」は、図2(a)の文字列入力領域52から関数51に向かう矢印58に対応するセル304を示すポイントが入っている。

【0167】図21の306、307は、それぞれ「文字カーソルのある文字列入力領域」およびその文字列入力領域内での「文字カーソル位置」を表わす共通変数である。図2(a)では、文字カーソルは文字列入力領域52内にある(図の53)。したがって、変数306すなわち「文字カーソルのある文字列入力領域」には、図2(a)の文字列入力領域52に対応するセル302指すポイントが入っている。また、文字カーソル自身は、文字列入力領域の1行目の4文字目(4文字目は次に文字を入力する位置)にあるので、変数307すなわち「文字カーソル位置」には、値(4、1)が入っている。

【0168】この状態でさらにキーボードから文字が入力された場合、図20のステップ211の処理(文字入力処理)により、変数「文字カーソル」位置に文字が挿入表示され、文字カーソルは一つ右に動く。

【0169】また、挿入、削除等の各種編集機能に対応するキー入力が行なわれた場合は、図20のステップ213において、文字の挿入、削除等の処理が行なわれる。

【0170】一方、図20のステップ210でキーボードから入力されたものが変換キーであると判定された場合は、ステップ212に進み、変換キーに対する処理を行なう。

【0171】図22および図23に、変換キーの処理の詳細を示す。

【0172】図からわかるように、変換キーが押されたときの状況によって、処理が様々に異なる。

【0173】まず、ステップ220で、変換キーが押されたとき、文字カーソルがどの文字領域にあるか判定する。文字カーソルが、関数等の実引数もしくは変数等の初期値に対応する文字入力領域にあるときは、ステップ221に進む。図2(a)のような場合がこれに相当する(文字カーソル53は、関数foo(図の51)の引数位置にある文字列入力領域52中にある)。また、図6の73のような、局所変数定義の初期値に対応する文字入力領域内に文字カーソルがある場合も、これに該当する。

【0174】次に、ステップ221では、その文字列入力領域に既に文字列が入力されているかどうか判定する。既に文字列が入力されている場合(図2(a)のような場合)、図式プログラムへの変換処理を行なう(ステップ222)。変換処理の詳細は後述する。

【0175】ステップ221で、文字カーソルの存在する文字列入力領域内に文字列が入力されていない場合、その文字列入力領域の種別を判別する(図22のステッ

プ223)。その文字列入力領域がその上位セルの省略可能な引数または変数等の初期値の場合、その文字列入力領域を削除する(ステップ224)。ここで、省略可能な引数または初期値とは、例えば前記ifの引数elseのようなもの(図8(b)の84)、あるいは図6のletの局所変数の初期値(図の73)である。また、前記letの変数リスト中の変数のような、任意個設定可能な引数の一つもこれに該当する。

【0176】ステップ223で、文字カーソルがある文字列入力領域が、省略可能でない引数に対応するもの場合は、何もしない。省略可能でない引数とは、必ず指定しなければならない引数、例えば関数remの引数number、divisorなどである。

【0177】ステップ223のあと、次にステップ225、226、227で、文字カーソルを次の文字入力領域に移動する処理を行なう。ステップ225で、注目している(該当する文字列入力領域を削除した場合には、注目していた)文字列入力領域に、後続する引数の有無を調べる。後続引数がある場合には、その引数の文字列入力領域に文字カーソルを移動する(ステップ226)。後続引数がない場合には、その文字列入力領域のより上位の言語要素の引数を再帰的に探し、その文字列入力領域より相対的に後方で最初に見つかった文字列入力領域に、文字カーソルを移動する(ステップ227)。

【0178】図22のステップ220で、文字カーソルのある文字列入力領域が変数リスト中の変数の一つ、すなわち、letの局所変数リスト、関数定義defunの仮引数リスト、またはdoの繰り返し制御変数リスト等の変数リスト中の変数の一つであったとき、次に、ステップ228で、その文字列入力領域に既に文字列が入力されているかどうか調べる。文字列が既に入力されているならば、変数名を指定文字列の内容に変更する(ステップ229)。その文字列入力領域に文字列がまだ入力されていないならば、前記ステップ224の場合と同様、その文字列入力領域を削除する(ステップ230)。

【0179】なお、図には示していないが、ステップ228、229、230の処理のあと、図23の225、226、227に示したような、次の文字列入力領域への文字カーソルの移動処理を行なう。

【0180】ステップ220で、文字カーソルのある文字列入力領域が、言語要素に対応する図形部品の名称表示領域、例えば、図2(b)の関数fooや関数remの名称を示す文字領域(59、60)、または、図12のブロック構造letの名称表示領域(71)、あるいは図8(a)の条件分岐ifの名称表示領域(80)などにあるとき、ステップ231で、その領域に表示されている文字列と図形部品の名称とを照合する。これが一致している場合には何もしない。異なっている場合に

は、図形部品に対する変更指示と解釈し、表示されている図形部品を、文字列で指定した名称を持つ図形部品に変更する（ステップ232）。このとき、図9の例で説明したように、変更前と変更後で引数に不一致があれば、不足している引数は補い、過剰な引数は強調表示して、ユーザに注意を促す。

【0181】変更後、新しい図形部品の第1引数に対応する文字列入力領域の先頭に、文字カーソルを移動する（ステップ233）。

【0182】次に、図22のステップの図式プログラムへの変換処理（ステップ222）の内容について説明する。

【0183】図24は、図1の図式プログラム部品表16の内容を示したものである。表の要素は、それぞれ使用言語（この場合はCommon LISP）の言語要素ごとに、名称、文字表現による構文、およびそれに対応する図式プログラム部品を表わす図形に関するデータからなっている。

【0184】なお、図24中の「・・・」は、任意個の同じものが繰り返されることを示している。例えば、項番1の「一般の関数」において、「文字表現の構文」の欄の<引数2>の後の「・・・」は、<引数3><引数4>・・・が必要なだけ続くことを表わしている。

「図式プログラム部品」の方も同様である。ただし、任意個であるから、何も続かなくてもよい。また、「一般の関数」で引数が一つの場合が示されていないが、ほぼ項番1の場合と同様であるので、記述を省略した。

【0185】図25に、図式プログラムへの変換処理222のフローを示す。

【0186】ステップ240で、まず、文字列入力領域中に入力されている文字列の内容を調べる。これが、関数、変数等の名称を表わす文字列であるとき、ステップ241に進み、次に、それが図式プログラム部品表16に登録されているものかどうか調べる。表中に一致するものが見つかった場合には、その文字列が表わす言語要素に対応する図形部品と、その各引数に対応する文字列入力領域を表示し、また、各引数の文字列入力領域内に、その引数の仮引数名称をイタリック体で表示する（図16ステップ242および243）。このとき、文字列から対応する図形部品その他への変換には、図1の文字表現から図式表現変換へのプログラム12の機能の一部を利用する。図2（b）の関数rem、図6のブロック構造let、図8（a）の条件分岐ifなどは、この処理によって表示されたものである。なお、ステップ243において、仮引数名称を、図2（b）のような引数に対応する文字列入力領域でなく、図3のように関数等の箱の中に表示するように指定されているときは、それにしたがって処理を行なう。

【0187】一方、ステップ241で、文字列が変換規則表6に登録されていない名称であるときは、その名称

はユーザ定義関数の名称であると判断する。そして、汎用の関数呼び出しの図形部品と、その一つの引数に対応する文字列入力領域を表示する（ステップ244）。図2（a）の関数fooは、この処理の結果表示されたものである。

【0188】ステップ241のあと、ステップ245に進み、新しく生成表示した図形部品の第1引数に対応する文字列入力領域の先頭に、文字カーソルを移す。

【0189】ステップ240で、文字列が（部分）プログラムを表わすものであると判定されたとき、図1の「文字表現から図式表現変換へのプログラム」12を利用して、その（部分）プログラムを図式表現プログラムに変換する（ステップ246）。

【0190】なお、上記の説明で、図25が「図式プログラムへの変換処理」であり、その中のステップ246の処理内容が「文字表現プログラムを図式表現に変換」であるので、両者は同じことを表わしているのではないか、という疑念を生じさせる恐れがある。しかし、両者には次のような相違がある。

【0191】まず、ステップ246の処理内容は、文字表現プログラムを図式表現プログラムに変換する。すなわち、ある形態の「プログラム」を、別の形態の「プログラム」に変換する処理である。

【0192】これに対し、図25全体の処理は、ステップ246の処理も含むが、それ以外の処理、すなわち、入力文字列として図式プログラム部品の名称が与えられたとき、これを対応する図式プログラム部品に変換する処理を含む（ステップ241、242、243、244、245）。

【0193】後者の処理は、（完全な）プログラムの変換処理でなく、プログラムの「極く一部」、すなわち、図式プログラム部品の「名称」を与えて、対応する図形部品を表示する処理である。この点、誤解なきようにお願いしたい。

【0194】なお、ステップ246の処理、すなわち図1の「文字表現から図式表現変換へのプログラム」12の詳細については、前述したように、先願（特願平2-056360）に記した方法を用いることができるので、ここでは説明を省略する。

【0195】前述した図4（a）から図4（b）への変換は、上記のステップ246の処理、すなわち、図1の「文字表現から図式表現変換へのプログラム」12の処理によって行なわれたものである。このとき与えられた文字列が図5（a）のように誤りを含むプログラムであるときは、図5（b）のように適宜不足の引数に対する文字列入力領域を追加し、また過剰な引数は強調表示を行なって、ユーザに注意を促すようにする。

【0196】次に、図19のマウス・ボタン処理203について説明する。

【0197】マウスは通常、画面上に表示されている図

形カーソルを操作して、画面上の対象の選択、コマンドを適用する場所の指定などに用いられる。

【0198】画面上に既に図式表現プログラムが表示されているとして、それを構成する図形部品の近傍に図形カーソルを移動し、マウス・ボタンを押下すると、その図形部品が選択される。選択された図形は、選択されたことをユーザに通知する意味で、反転表示などによる強調表示がなされる。この後、選択された図形部品に対してカット（切り取り）、移動、複写、削除などのコマンドが実行可能である。

【0199】また、一つの文字入力領域内に図形カーソルを移動し、マウス・ボタンを押下すると、その文字入力領域内の図形カーソルによる指定位置に、文字カーソルが移動する。

【0200】次に、図19のコマンド処理204について説明する。

【0201】図26はその詳細フローである。

【0202】まず、ステップ250で、コマンドの種別を判定する。コマンドが挿入コマンドならばステップ251へ、ペースト・コマンドならばステップ252へ、移動コマンドならばステップ253へ、複写コマンドならばステップ254へ、削除コマンドならばステップ255へ、逆変換コマンド、すなわち画面上で指定した部分的な図式プログラムを文字プログラムに（逆）変換するコマンドならば、ステップ256へ、それぞれ進む。

【0203】図27、図28に、挿入コマンドの処理フローを示す。

【0204】ここで、図27、図28の説明に入る前に、前記図11～図14で説明した挿入可能位置の強調表示方法について説明する。

【0205】挿入可能位置に図形カーソルが来たときその部分を強調表示するためには、まずその時の図形カーソルがある位置の座標を読み取る。次に、これをその場所にある図形の座標と照合する。これがある誤差内にあり、またその図形のその位置が挿入可能位置であると判定されたならば、その部分を強調表示する。このとき、図形の該当部分が挿入可能位置であるかどうかの判定方法が問題であるが、これは次のようにすれば簡単に行なうことができる。

【0206】図11～図14で示したような強調表示を行なっている部分に対応する（挿入可能位置表示図形）を、あらかじめ画面上に表示されている全ての図形部品の挿入可能位置に対応して作成しておく。ただし、通常は、ユーザに見えない形、すなわち内部データとしてのみ生成しておく。そして、挿入コマンドが起動され挿入コマンド・モードに入ったとき、図形カーソルの位置座標と、この挿入可能位置表示図形の座標と比較し、ある誤差内にあれば、この図形を画面に表示する。また、図形カーソルが図形から離れていったときは、この図形を見えない状態に戻す。

【0207】図12で示した例を例題にとって具体的に説明する。図29は、図12をほぼ再録した図である。ただし、図12にある要素に加え、二つの挿入可能位置表示図116、118が追加されている。また、図12で太線で表示されていた挿入可能位置表示図形114は、図形116、118と同じく点線で表示されている。図形114、116、118は、上記の見えない図形を模式的に示したものであり、図29で点線で表示しているのは、説明の便宜のためである。同じく説明の都合上、図形114、116、118の存在位置を図形110の右辺から右方向に少々ずらしてあるが、これらの図形は、実際は、関数 `fun1` を表わす図形110の右辺に重畳する位置にある。これら三つの図形は、それぞれの位置にプログラムを挿入可能であることを示したものである。

【0208】今この状態で図形カーソルが図形114の上に来たとする。そうすると、上記の説明にしたがい、図形114が顕在化され、図12のように図形114が太い実線で表示される。

【0209】図形116、118についても同様で、図形カーソルがどちらかの上に来ると対応する図形が顕在化されて太線表示となり、その位置にプログラムが挿入可能であることを示す。なお、図形116、118は、それぞれ、関数 `fun1` の第1引数の前、および第2引数の後に挿入可能なことを示すために、予め準備されている。

【0210】図30は、図29に示した図式プログラムの内部データを示したものである。

【0211】図30の310、311、312は、それぞれ図30の関数 `fun1`、`fun2`、`fun3` に対応するセルである。図21のセル300と比べて、それぞれのセルには、スロット「挿入可能位置表示図形」が新たに加わっている。これは図21では簡単化のために省略した。また、図を簡単にするために、スロット「名称」には名称を表わす文字列を直接記述している。また枠を表わすセルは簡単化のために図示していない。

【0212】図の316～318は、図29の挿入可能位置を表示するための図形114、116、118にそれぞれ対応するセルである。通常のセルの持つ「分類」「位置」「サイズ」の各スロットに加え、固有スロットとしてスロット「表示」を持つ。このスロット「表示」は、「表示」または「非表示」を値として持つ。通常本スロットの値は「非表示」となっているが、図形カーソルがこれらの図形の近傍に来たとき、すなわち図形カーソルの位置座標が、スロット「位置」と「サイズ」で示される領域とある一定誤差内で重なっているとき、スロットの値は「表示」に変わる。このとき、図1の入力表示制御プログラム6により、対応する図形が画面に表示される。

【0213】ところで、図30において、関数等の戻り

値を表わす矢印の図形に対応するセル314～315には、スロット「表示」が新たに加わっている。図11

(a)で明らかなように、矢印の場合、その上に図形カーソルが来たとき、矢印全体が強調表示される。したがって、矢印の場合は挿入可能位置表示図形を別に設けずに、その機能を矢印の属性とした。この「表示」スロットの値は、上記とは異なり、「通常表示」または「強調表示」である。図形カーソルが矢印上に来たとき、スロット「表示」の値は「強調表示」となり、図形カーソルがそこから離れたとき「通常表示」に戻る。

【0214】なお、セル320～324は、図21のセル305と同様に、複数の要素をまとめて一つのスロットにつなげるために用いられるセルである。

【0215】以上の方法により、図形カーソルが挿入可能位置を示す図形の上に来たときそこが強調表示され、図形カーソルがそこから離れると、その図形の強調表示は解除される。

【0216】なお、この方法を利用して、挿入可能でない位置でマウス・ボタンを押下したときのエラー検知も容易に行なうことが出来る。これは次のようにして行なう。画面上に表示されている「表示」スロットを持つセルのうち、その値が「表示」ないし「強調表示」になっているものを調べる。これが一つもないとき、図形カーソルは挿入可能位置以外のところを指しているわけで、このときマウス・ボタンが押されたらエラーであることがわかる。

【0217】以上の挿入可能位置の強調表示の方法は、上記の挿入コマンドの場合や、後述のペースト、移動、複写コマンドなどの場合に加えて、一般に場所に依存する属性を、図形カーソルがそこに来たときだけ表示するような場合にも使用することが出来る。

【0218】また、以上に示した方法は、前述した図式プログラム部品単位での挿入（または、ペースト、移動、複写）可能な位置全ての強調表示にも、さらに、画面上に表示されている図式プログラム中の全ての図式プログラム部品のうち、コマンド適用可能な部分を全て強調表示する方法にも、容易に拡張可能である。

【0219】以上で、挿入可能位置強調表示法の基本説明を終わり、図27、図28の挿入コマンドの処理フローの説明に戻る。

【0220】挿入コマンドが指示されると、まず最初にキーボード、マウスからの入力モードを挿入モードにする（図27のステップ260）。挿入モードに入ると、コマンドの実行が終了するまで、マウス以外からの入力を受け付けない。すなわち、マウスによる位置指定入力と、マウス・ボタン押下の入力のみを受け付けるモードである。また、このとき同時に、画面上に表示している図形カーソルを、挿入モードを示す形に変え、ユーザに現在挿入モードにあることを知らせる。

【0221】その後、マウス・ボタンが押されるまで、

図形カーソルの位置に応じた挿入可能位置の強調表示および解除を繰り返す（ステップ261）。

【0222】繰り返しの最初で、まず図形カーソルの位置を読み取る（ステップ262）。図形カーソルの表示位置はマウスによって制御されるが、これは図1の入力表示制御プログラム6内で行なわれる。ここでは、入力表示制御プログラム6に対して問い合わせを行ない、図形カーソルの位置座標値を得る。

【0223】次に読み取った図形カーソルの位置座標が、上記で説明した挿入可能位置を表示するための図形の近傍にあるかどうか判定する（ステップ263）。近傍にある場合には、次に、以前に（強調）表示されている挿入可能位置図形の有無を判定する（ステップ264）。（強調）表示されている挿入可能位置表示図形がある場合は、さらに、その（強調）表示されている図形が図形カーソル下の図形と同一か否かを判定する（ステップ265）。両者が一致している場合は何もしない（つまり（強調）表示をそのまま維持する）。両者が異なる場合には、以前（強調）表示している挿入可能位置表示図形の（強調）表示を解除し（ステップ266）、図形カーソルが指示している挿入可能位置表示図形を（強調）表示する（ステップ267）。

【0224】ステップ264で、以前に（強調）表示されている挿入可能位置図形がないことがわかったときは、単に図形カーソル下の挿入可能位置表示図形を（強調）表示する（ステップ268）。

【0225】ステップ263で、図形カーソルが挿入可能位置表示図形の上にないと判定されたときは、次に、以前に（強調）表示されている挿入可能位置表示図形の有無を判定し（ステップ269）、ある場合にはその（強調）表示を解除する（ステップ270）。

【0226】以上の処理を、マウス・ボタンが押されるまで繰り返す。

【0227】マウス・ボタンの押下が検出されたことが検知されたとき、ステップ261のループを脱出し、ステップ271に進む。ステップ271では、（強調）表示されている挿入可能位置表示図形の有無を判定する。

（強調）表示されている挿入可能位置表示図形があれば、その図形で指示される位置に対して、文字列入力領域の挿入処理を実行する（ステップ272）。（強調）表示されている挿入可能位置表示図形がないときは、挿入不可を示す警告音、警告メッセージを発する（ステップ273）。

【0228】最後に、挿入モードを解除し、キーボード、マウスからの入力を通常モードに戻す。また、図形カーソルを通常の形にする（ステップ274）。

【0229】次に、上記図33ステップ272の挿入処理の詳細について説明する。

【0230】図11～図14で説明したように、挿入の対象となる図形部品は、挿入される位置によって異なる

る。挿入されるべき部品の種別の判定は、図31の表を用いて、次のようにして行なう。

【0231】図31の表は、挿入位置と、挿入されるべき部品との対応を示したものである。表で、挿入位置は、図形部品の種別と図形部品のセルのスロット名称で示されている。後者の図形部品のセルのスロット名称は、図形部品の挿入可能部分の位置に対応したものである。

【0232】前記の挿入可能位置の強調表示機能により、挿入コマンド・モード中でマウス・ボタンが押されたとき、挿入可能位置表示図形が強調表示されている。そこで、まずこの図形に対する内部データのセルを取り出し、次に、このセルからその上位の図形部品のセルを取り出す。ここで、図30では、挿入可能位置表示図形とそれが属する図形部品との間のポインタの向きは逆であるので、このままでは上位の図形部品を取り出すのは簡単ではない。しかし、図30に示しているような図式表現プログラムの内部データは、基本的に木構造をなしているので、この木構造を頭からたどっていけば、(強調)表示されている挿入可能位置表示図形の上位の図形部品を見つけることは容易にできる。あるいは、セルのスロットを拡張し、挿入可能位置表示図形からその上位の図形部品へ向けての逆ポインタを張っておいてもよい。

【0233】次に、注目している挿入可能位置表示図形が、上位の図形部品のセルのどのスロットに接続されているかを調べる。そして、得られたスロットの名称と、上位図形部品の種別の情報を用いて、図31を参照し、挿入すべき図形部品の種別を決定する。

【0234】ところで、図式表現プログラム上で挿入処理を行なうためには、挿入可能位置表示図形で指定される位置が、具体的に上位の図形部品のどの位置に該当するかを知る必要がある。

【0235】図11のような例の場合は、挿入可能位置表示図形は関数等の戻り値を示す矢印であり、通常この矢印は一つしかないので、挿入位置はこの矢印がある場所であることがわかる。しかし、図12のような場合、挿入位置は関数の引数の位置であるとは言っても、さらに、第1引数の前、第1引数と第2引数との間、あるいは第2引数の後ろ、の三つの可能性がある。このどれであるかを判別するには、次のようにする。

【0236】図29の挿入可能位置表示図形114は、図30の図式表現プログラムの内部データ中で、セル317に対応する。セル317は、前後のセル316、318とともに、上位図形部品(関数fun1)のセル310のスロット「挿入可能位置表示図形」に接続されている。これらのセルの並びは、セルの空間的な並びの順序に対応させている。そこで、これらの同一グループに属するセルの中で、注目しているセルがどの位置にあるかを調べることににより、具体的な挿入位置を明らかにす

ることができる。

【0237】セル316～318に関して言えば、セル316は図29の挿入可能位置表示図形116に対応しているので、第1引数の前の位置を、セル317は図29の挿入可能位置表示図形114に対応しているので、第1引数と第2引数との間の位置を、また、セル318は図29の挿入可能位置表示図形118に対応している。したがって、図12(a)の例に対しては、第1引数と第2引数との間の位置に対して挿入処理を施せばよいことが、これでわかる。

【0238】以上で求めた挿入部品の種別、および挿入位置の情報を用いて、文字列入力領域の挿入処理を実行する。

【0239】上記では、主に図11および図12の例を用いて説明したが、図13および図14についても、同様の方法を用いて挿入処理を実行できる。

【0240】次に、ペースト・コマンド(図26のステップ252)の処理のについて説明する。

【0241】ペースト・コマンドは、前述したように、挿入形式もしくは置き換え形式によりコマンドを実行することが出来る。どちらにしろ、コマンド実行可能位置の強調表示機能、およびコマンド実行不可の位置でコマンドの実行を指示したときのエラー通知機能がある。これらは、上記の挿入コマンドの場合に準じた方法を用いて実行することが出来る。

【0242】挿入形式によるペースト・コマンド実行可能位置に図形カーソルを持ってきたときは、前記の挿入コマンドの場合と同様、図形カーソル位置に挿入可能位置表示図形があるかどうか調べる。あるならば、その挿入可能位置表示図形がつながれている上位の図形部品のスロットをチェックする。このスロットと、ペーストの対象となっている図式プログラムの最上位の図形部品を照合し、そのスロットにペースト対象の図形部品をつなぐことが出来ることがわかれば、注目している挿入可能位置表示図形を(強調)表示する。

【0243】この接続可否のチェックは、スロットに接続可能な対象の属性を付加しておく、あるいは、別途、スロット名称と接続可能対象の対照表を用意しておくことなどにより、容易に行なうことが出来る。

【0244】置き換え形式によるペースト・コマンド実行可能位置に図形カーソルを持ってきたときは、図形カーソル下にある図形部品の上位の図形部品のスロットと、ペースト対象となっている図式プログラムの最上位の部品とを照合する。そのスロットにペースト対象の図形部品をつなぐことが出来ることがわかれば、図形カーソル下の図形部品を(強調)表示する。なお、この(強調)表示のために、各図形部品のセルには、それを表現するためのスロット(例えば「表示」)が必要である。また、接続可否のチェックは、上記挿入形式における接

続チェックとまったく同様にして行なうことができる。

【0245】ペースト可能でない位置でマウス・ボタンを押下したときのエラー検知も、挿入コマンドの場合とほぼ同様にして行なうことができる。すなわち、画面上のペースト・コマンドにより（強調）表示されているものの有無を調べればよい。（強調）表示されているものがない場合には、図形カーソルはペースト可能位置以外のところを指しているわけで、このときマウス・ボタンが押されたらエラーであることがわかる。

【0246】ペースト・コマンドの具体的な実行も、挿入コマンドの場合に準じる。すなわち、挿入形式による実行の場合は、前記挿入コマンドの場合と同様にして具体的に挿入位置を求め、その位置に、退避領域に退避している図式プログラムを挿入する。置き換え形式では、図形カーソル下の上位の図形部品のセルの該当スロットの内容を、退避領域に退避している図式プログラムで置き換える。

【0247】さて、次に、図26のステップ255の削除コマンドの処理について説明する。削除コマンドを実行するには、あらかじめマウス等の位置指定装置により、削除したい図形部品を選択しておく必要がある。次に削除コマンドが指定されると、選択され強調表示されている図形部品のセルを求め、さらに、その上位セルを求める。上位セルの該当スロットから選択されている図形部品のセルを外す。また、この図形部品が引数を持つ場合には、この引数に対応する図形部品のセルを選択されている図形部品のセルの該当スロットから外す。そして、外した図形部品のセルを、選択された図形部品のセルの上位のセルの、今まで選択された図形部品が接続されていたスロットにつなぐ。このとき、選択された図形部品が複数の引数を持つ場合には、一つを除いた残りの引数は一般に過剰な引数となるので、これらを上位図形部品のセルの「余分な引数」のスロットにつなぐ。以上の内部データに対する処理が終わったあと、このデータに基づき、画面上に表示されている図式プログラムを修正する。なお、上記の「余分な引数」のスロットにつながれている図式プログラムは、過剰な引数であることをユーザに通知するため、強調表示を行なう。また、選択された図形部品が、下位に引数をもたないときは、上位の図形部品のセルに新たに文字列入力領域のセルを接続する。

【0248】最後に、図26のステップ256の図式プログラムから文字プログラムへの逆変換の処理である。

【0249】これは、あらかじめ選択されている図式プログラムに対するセルを取り出し、図1の図式表現から文字表現への逆変換プログラム14に処理を依頼すればよい。

【0250】なお、逆変換プログラム14の処理しては、先願（特願平2-056360）に記した方法を用いることができる。したがって、ここでは処理の詳細に

についての説明は省略する。

【0251】以上、各種の入力編集方法の具体的実現方法について説明した。

【0252】

【発明の効果】以上説明してきたように、本発明によれば、図を文字だけすなわちキーボードのみにより入力可能になる。これによって、図形部品が多数ある場合にも、メニュー上の多数の項目の中から所望のものを探すという面倒な操作なしに、図形をその名前を用いて直接画面に表示することができる。

【0253】また、本発明によれば、図形で表現したプログラムを、やはり、文字すなわちキーボードのみで、特別な図形指定を行なうことなくしに、入力することが出来る。これによって、従来の図式表現プログラムの作成上のネックであった入力の問題を解消し、文字表現プログラムと同等の入力効率で、プログラムの入力を行なうことが出来るようになった。

【0254】さらに、プログラムの入力にあたっては、プログラム言語の要素の名称を入力すると、その言語要素に対応する図式表現と付随する引数が合わせて示されるので、ユーザは、言語要素の持つ引数の種類および順序が不確かなものであっても、いちいちマニュアル等で確認することなくプログラムを入力することが出来る。当然、引数の種類および順序に関する思い違いに起因するプログラムの誤りは、本発明による図式プログラム入力編集方法を用いることによって、まったく無くすることが出来る。

【0255】また、新しいプログラムの挿入、あらかじめカット（切り取り）コマンド等によって退避領域に退避しているプログラムのペースト（貼り付け）、また移動、複写などのコマンドの実行にあたり、コマンドを実行可能な位置を画面上で表示してくれるので、間違った位置で上記のコマンドを実行するということがない。

【0256】削除においても、その結果プログラムの構文が不適切なものになった場合には、誤っている箇所を指示してくれるので、構文が間違っただけのままプログラムの編集を終了してしまうというのを防いでくれる。

【0257】このように、本発明による方法を用いることにより、わかりやすい図式によるプログラム表現を用いて、効率よく、かつ構文的な誤りのないプログラムの作成を行なうことが出来る。

【0258】なお、これまでの図式プログラムの入力編集方法の説明では、既存の文字表現によるプログラム言語によるプログラムを図式的に表現した例を用いて説明してきた。しかし、本発明は、最初に述べたように、文字による表現形式を持たない図式のみによるプログラム言語、いわゆる視覚言語にも適用可能であることは、言うまでもない。さらに、図式表現によるプログラムだけでなく、電子回路図や化学プラント配管図などの木構造もしくはネットワーク構造により表現される各種図面に

も容易に適用可能であり、それら図面の入力編集を容易にし、作成した図面の構文的な誤りを防止することにも効果がある。

【図面の簡単な説明】

【図1】本発明による図式プログラム入力編集方法のシステム構成を示す図である。

【図2】本発明に基づく図式プログラムの入力例を示す図である。

【図3】仮引数の別の表示方法を示す図である。

【図4】文字プログラムの図式プログラムへの変換例を示す図である。

【図5】誤りを含む文字プログラムの図式プログラムへの変換例を示す図である。

【図6】本発明に基づく図式プログラムの入力例を示す図である。

【図7】任意個設定可能な引数の自動追加の例を示す図である。

【図8】本発明に基づく図式プログラムの入力例を示す図である。

【図9】名称変更による図形部品の変更の例を示す図である。

【図10】名称変更による図形部品の変更の他の例を示す図である。

【図11】文字列入力領域の挿入例を示す図である。

【図12】文字列入力領域の挿入例を示す図である。

【図13】文字列入力領域の挿入例を示す図である。

【図14】文字列入力領域の挿入例を示す図である。

【図15】置き換え形式によるペースト・コマンド実行例を示す図である。

【図16】図式プログラムの部分的削除の例を示す図である。

【図17】本発明による方法の別の図式表現(PAD)への適用例を示す図である。

【図18】本発明による方法の別の図式表現(PAD)への他の適用例を示す図である。

【図19】入力編集プログラムのフローを示す図である。

【図20】キーボード入力処理のフローを示す図である。

る。

【図21】図式表現プログラムの内部データ表現の例を示す図である。

【図22】変換キー処理の一部のフローを示す図である。

【図23】変換キー処理の本来のフローを示す図である。

【図24】図式プログラム部品表の内容を示す図である。

【図25】図式プログラムへの変換処理のフローを示す図である。

【図26】コマンド処理のフローを示す図である。

【図27】挿入コマンド処理の一部のフローを示す図である。

【図28】挿入コマンド処理の他の部分のフローを示す図である。

【図29】挿入可能位置の強調表示方法を説明するための概念図である。

【図30】図式プログラムの他の内部データ表現を示す図である。

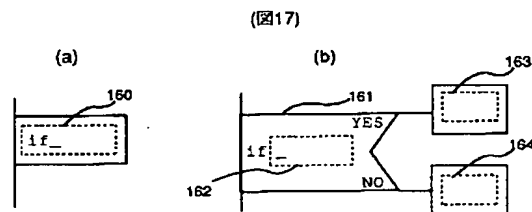
【図31】文字列入力領域の挿入位置と挿入される部品との対応を示す表である。

【図32】従来のプログラムを示す図である。

【符号の説明】

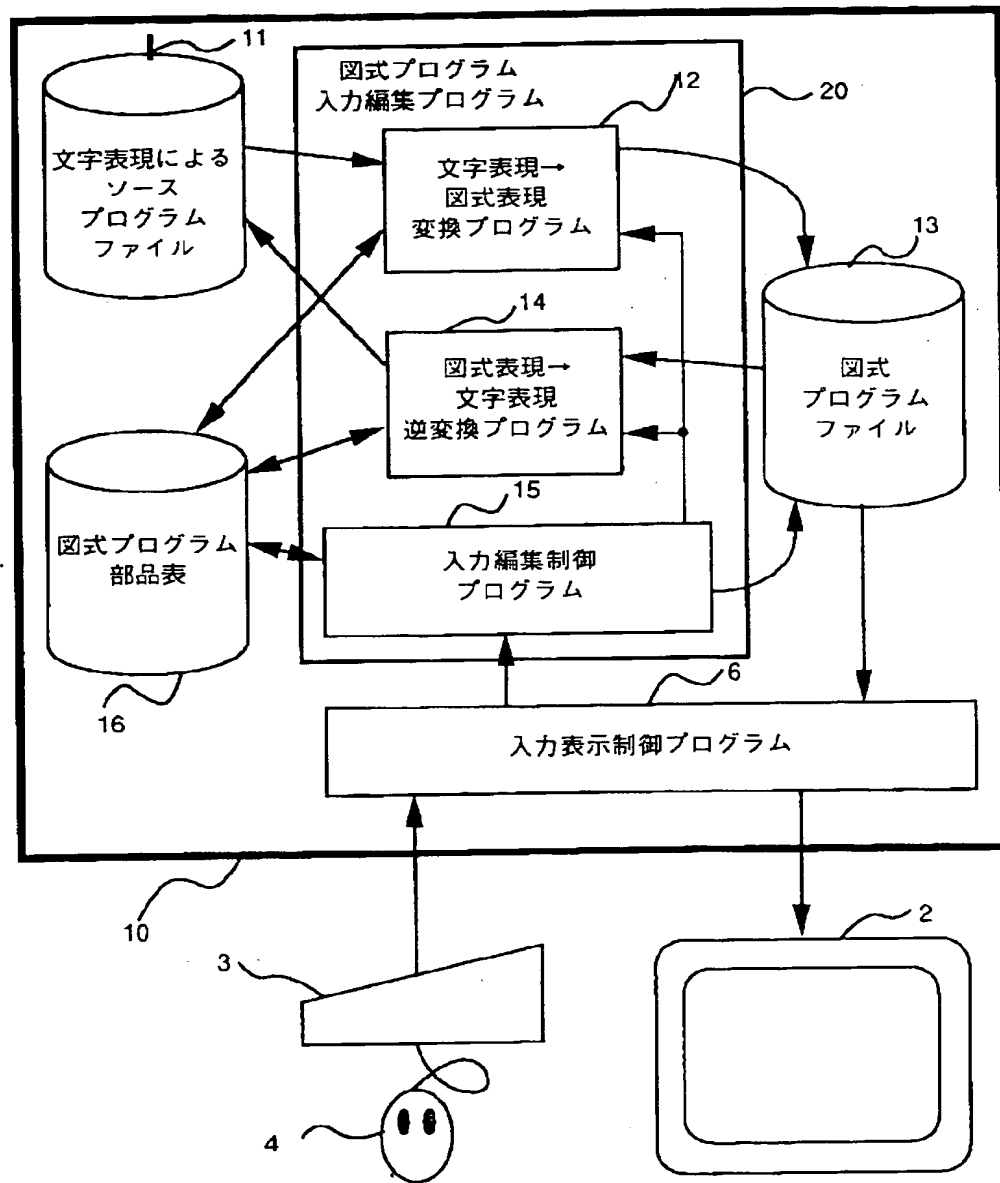
1…パーソナル・コンピュータ、ワークステーション等の制御装置、2…ディスプレイ装置、3…キーボード、4…マウス等の位置指示装置、5…図形データ・ファイル、6…入力表示制御プログラム、7…図形入力編集プログラム、8…図形入力プログラム、9…名称図形対応表、10…パーソナル・コンピュータ、ワークステーション等の制御装置、11…文字表現によるソース・プログラム・ファイル、12…文字表現から図式表現への変換プログラム、13…図式表現プログラム・ファイル、14…図式表現から文字表現への逆変換プログラム、15…図式表現プログラムの入力編集プログラム、16…図式プログラム部品表、501～537…各図のタイトル。

【図17】



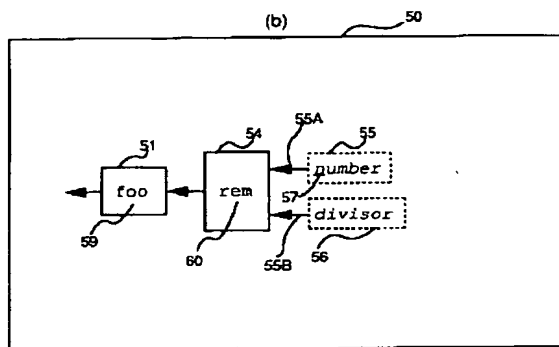
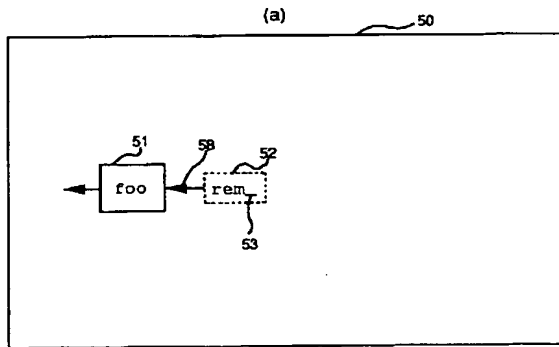
【図1】

(図1)



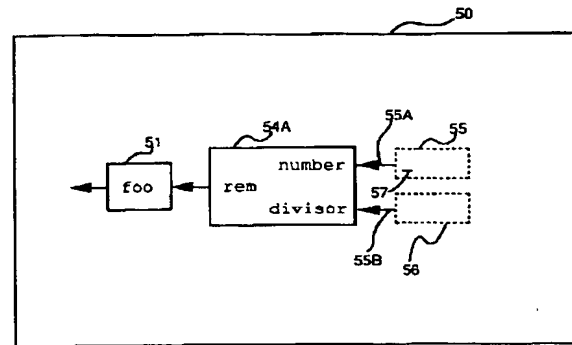
【図 2】

(図2)



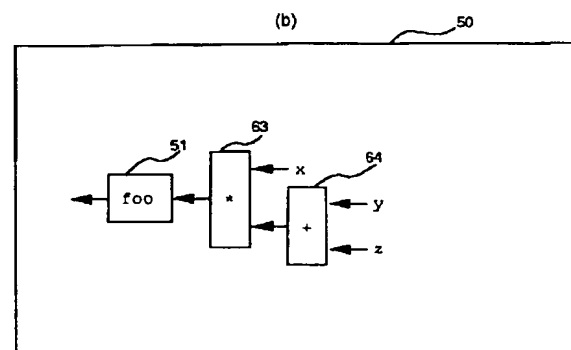
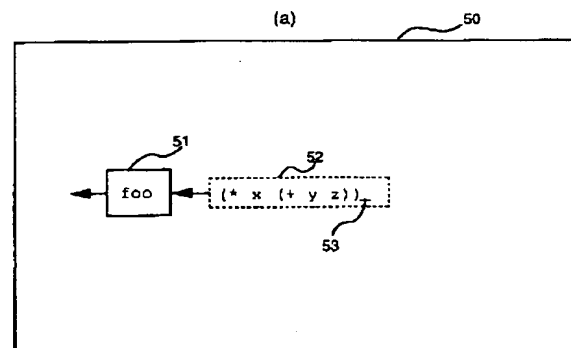
【図 3】

(図3)



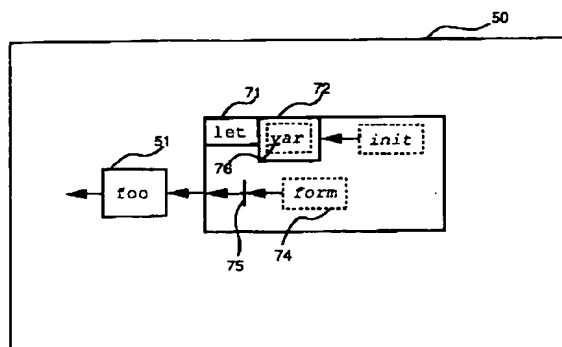
【図 4】

(図4)



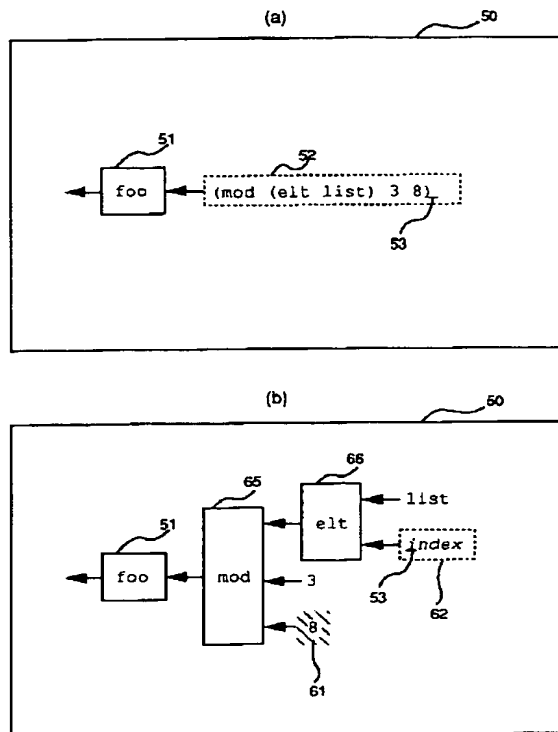
【図 6】

(図6)



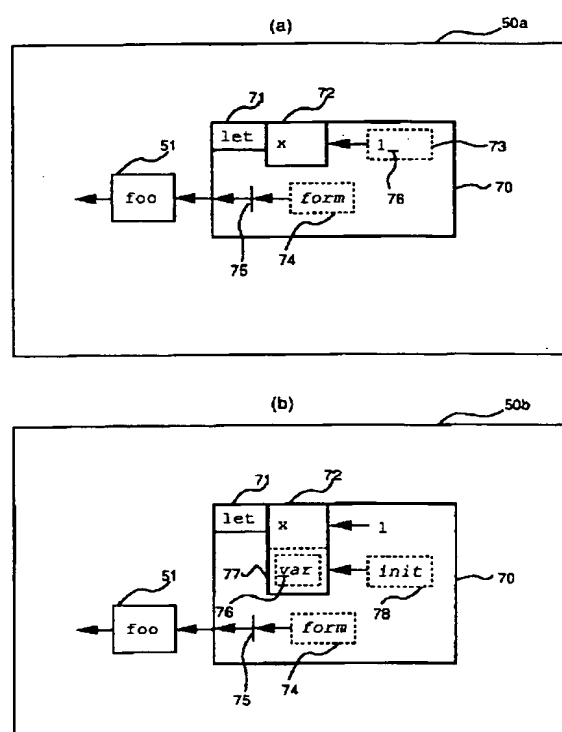
【図5】

(図5)



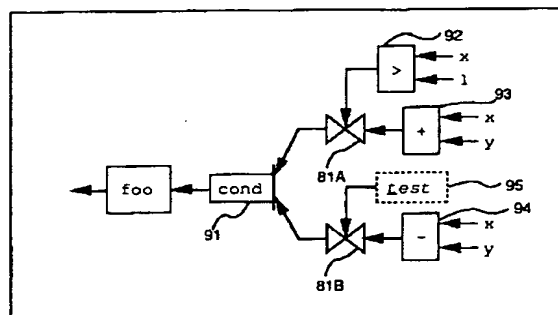
【図7】

(図7)



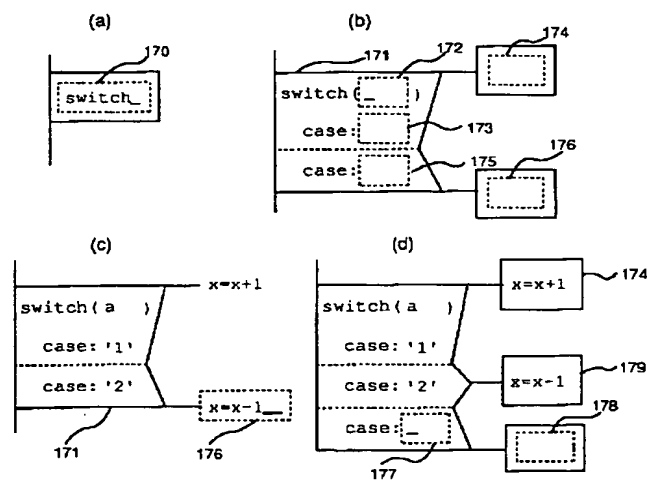
【図10】

(図10)



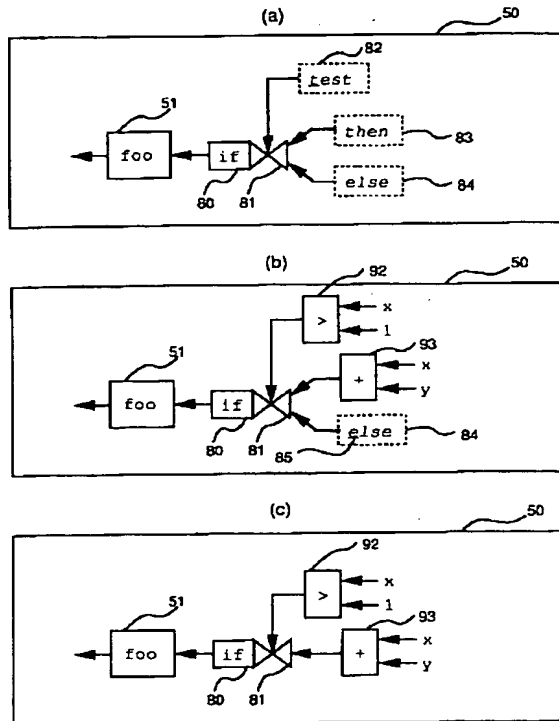
【図18】

(図18)



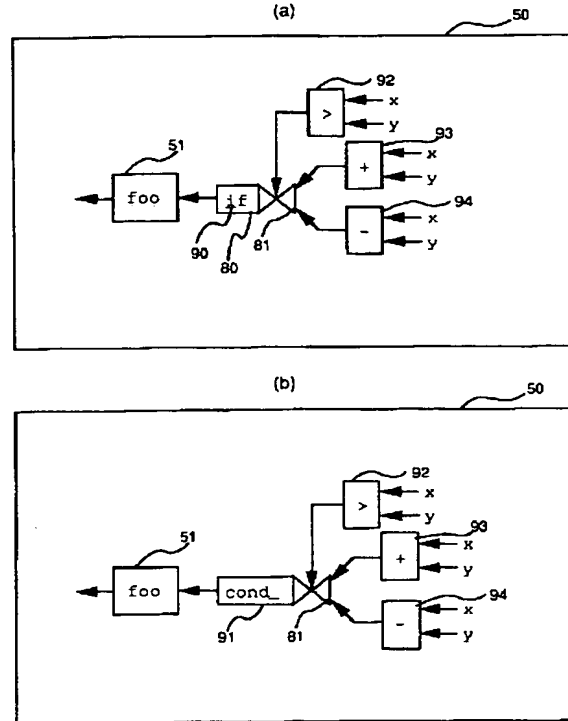
【図 8】

(図8)



【図 9】

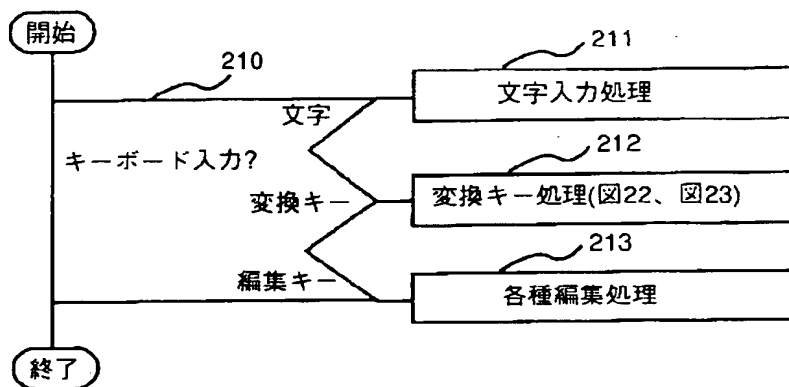
(図9)



【図 20】

(図20)

202 キーボード入力処理



【図 32】

(図32)

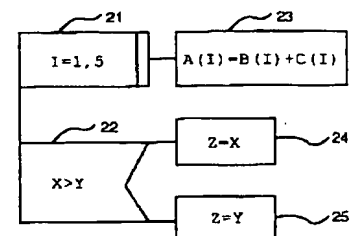
(a)

```

DO 10 I=1,5
  A(I)=B(I)+C(I)
10 CONTINUE
IF (X.GT.Y) THEN
  Z=X
ELSE
  Z=Y

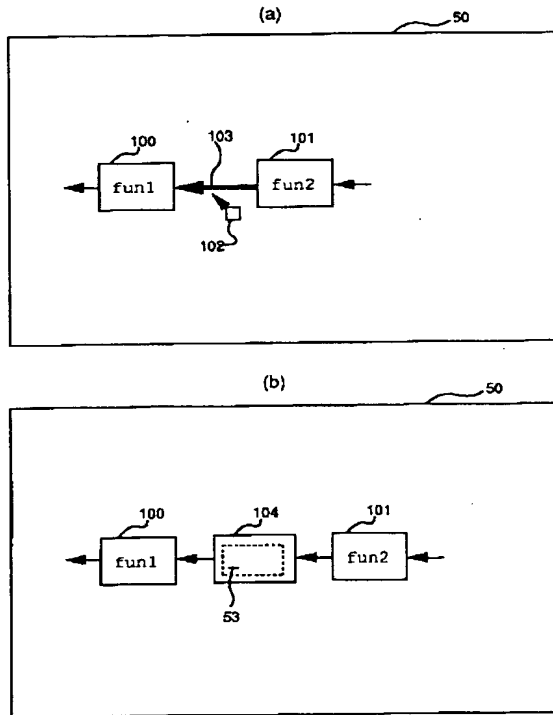
```

(b)



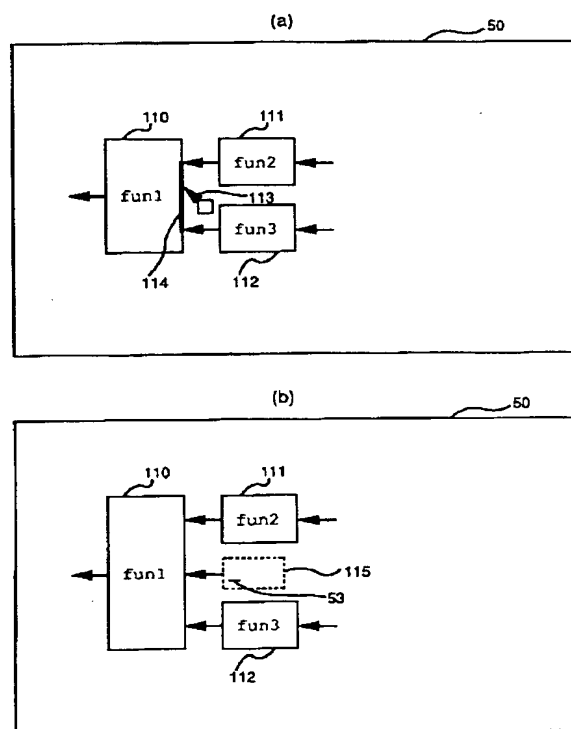
【図11】

(図11)



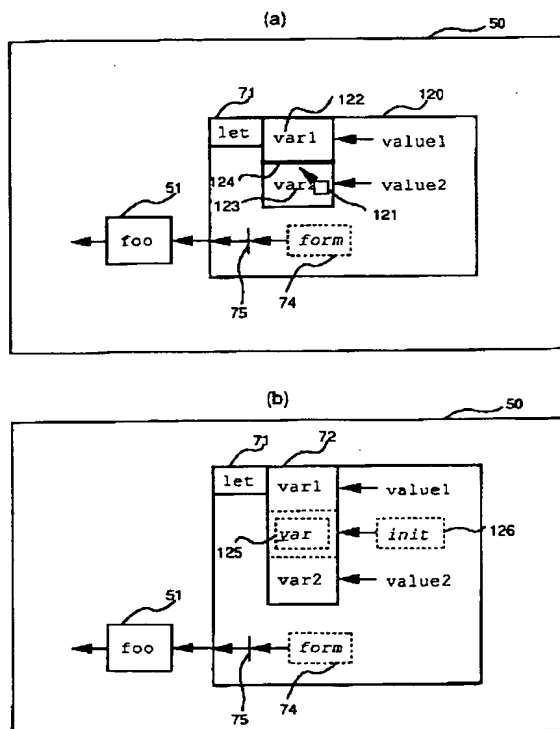
【図12】

(図12)



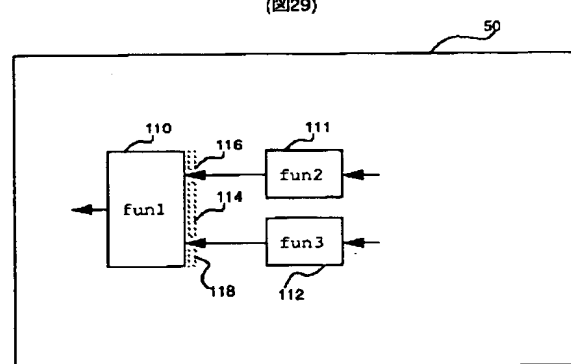
【図13】

(図13)



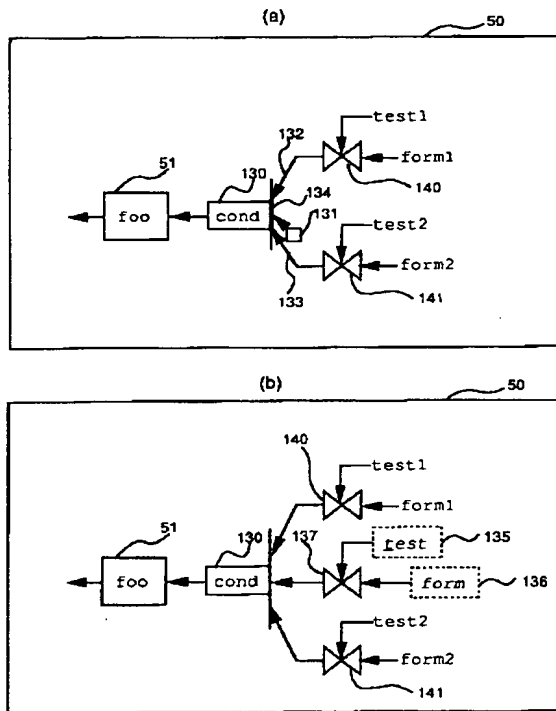
【図29】

(図29)



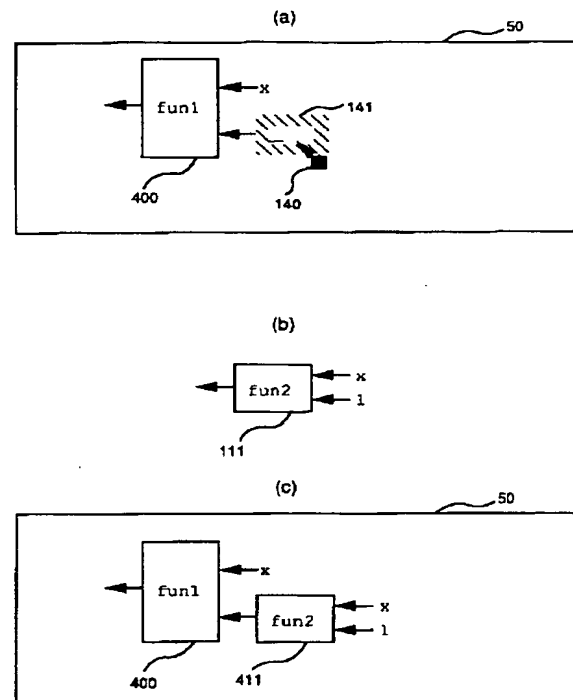
【図14】

(図14)



【図15】

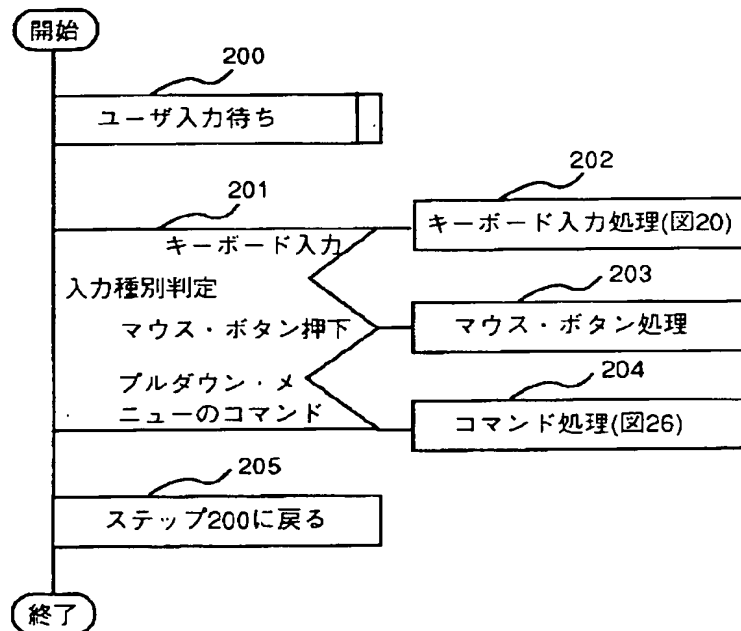
(図15)



【図19】

(図19)

15 入力編集制御プログラム

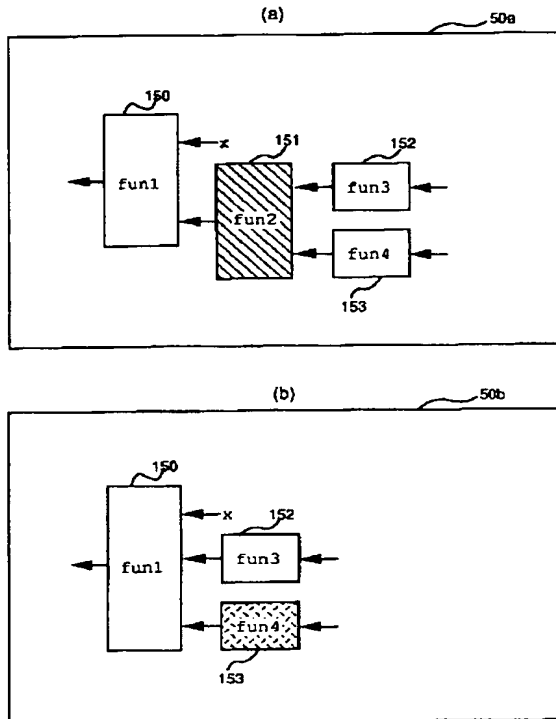


【図16】

【図23】

(図16)

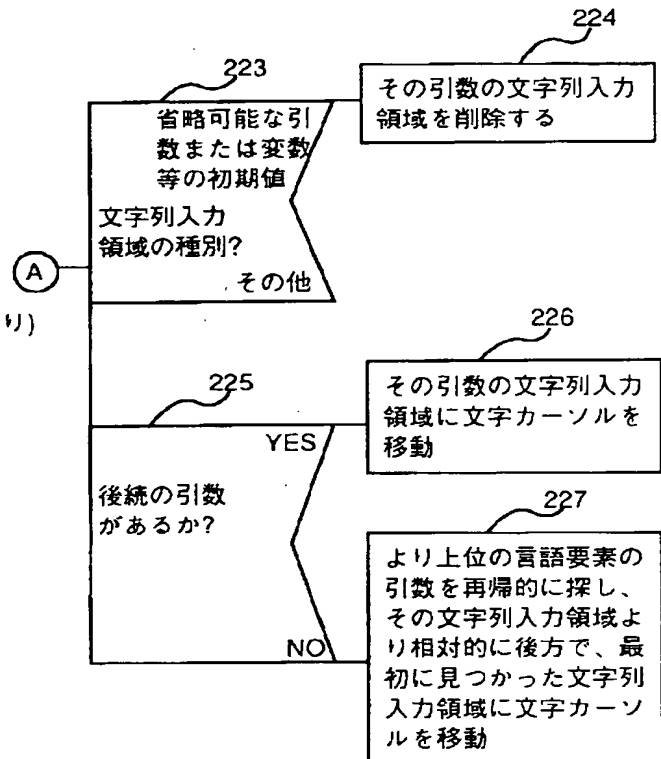
522 図式プログラムの部分的削除の例



(図22より)

(図23)

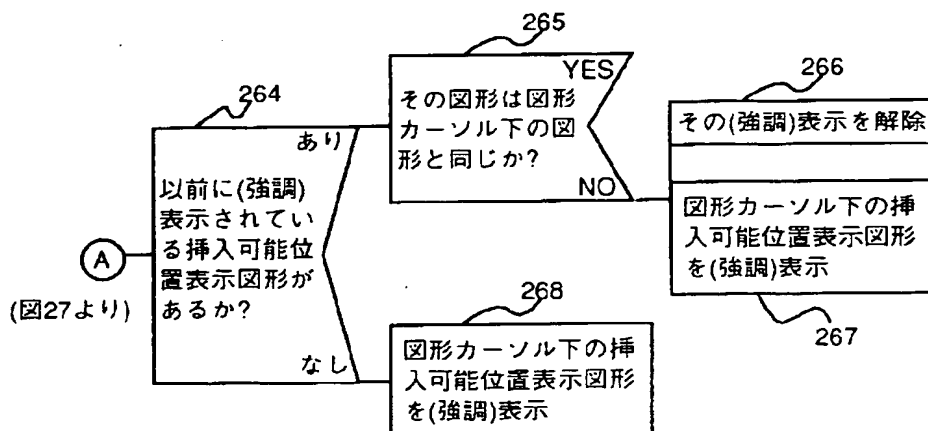
212 変換キーの処理(その2)



【図28】

(図28)

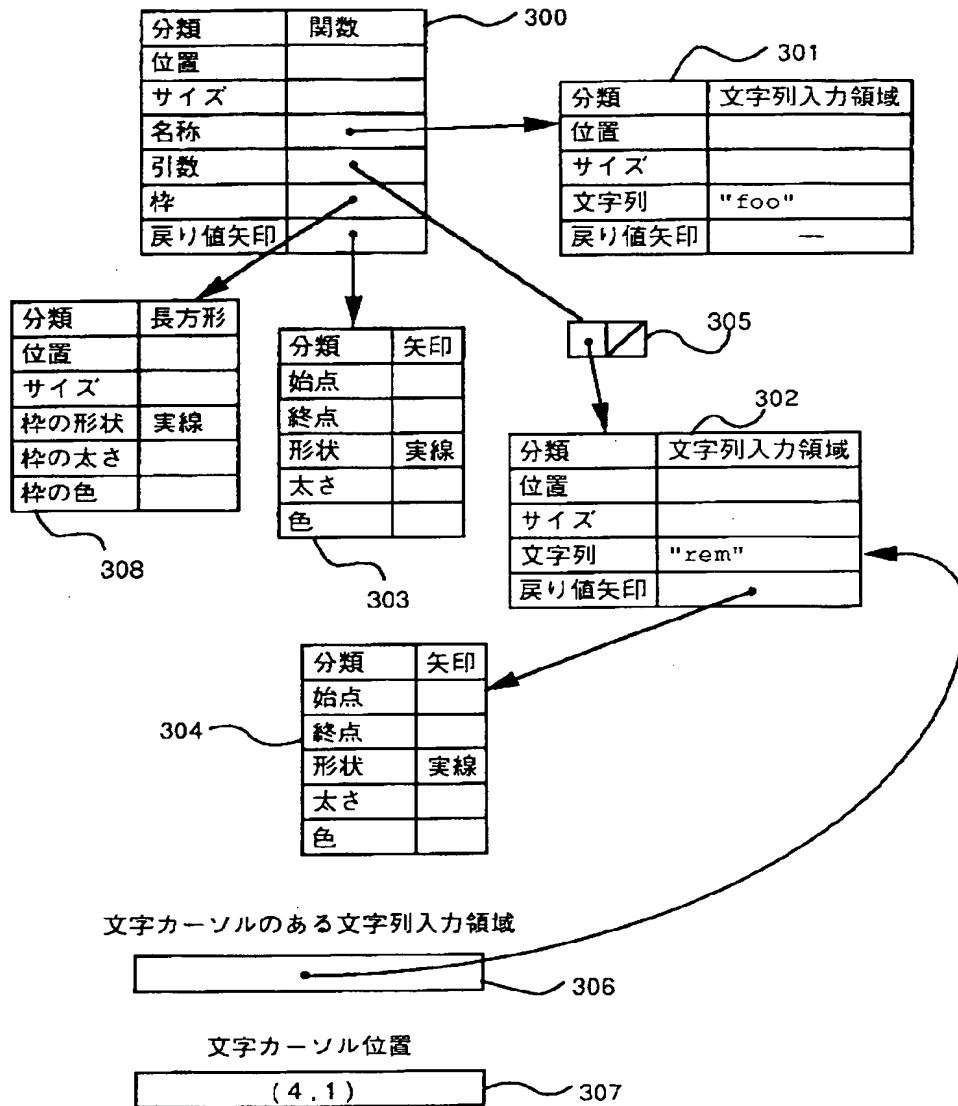
251 挿入コマンド処理(その2)



【図21】

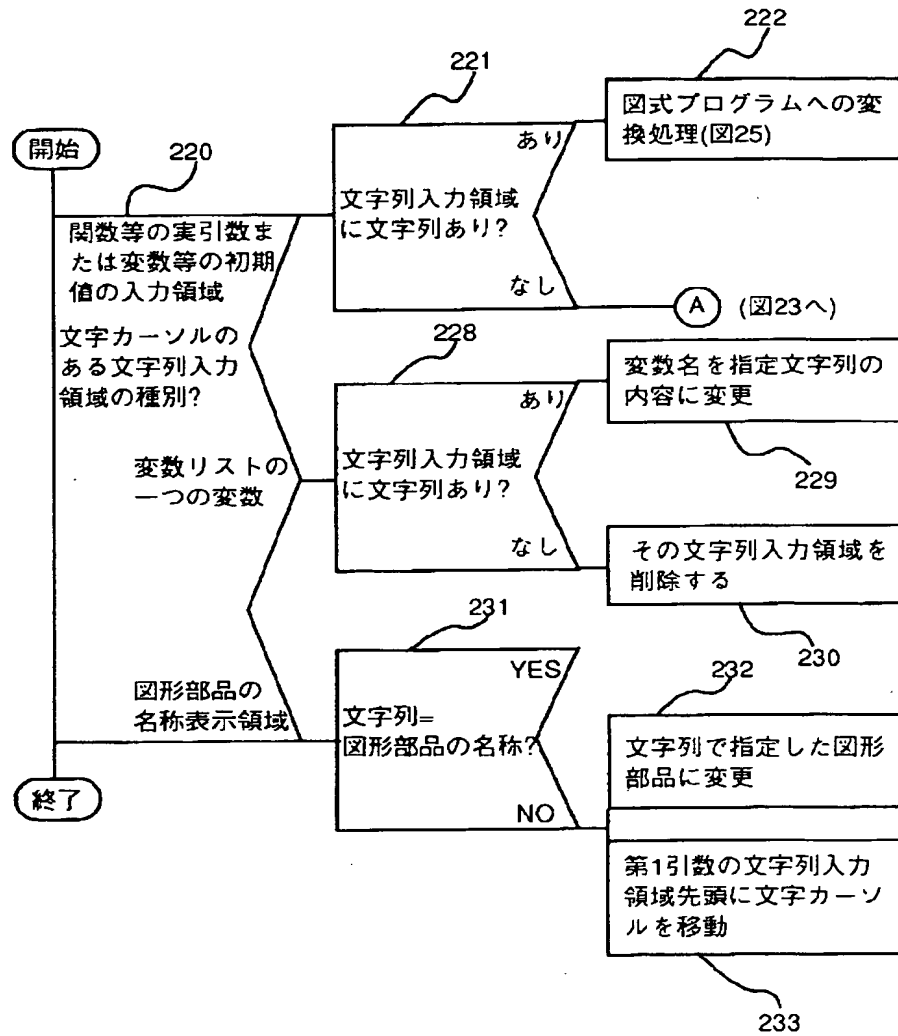
(図21)

527 図式表現プログラムの内部データ



【図22】

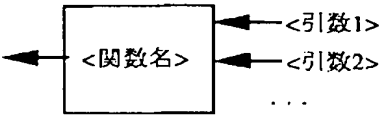
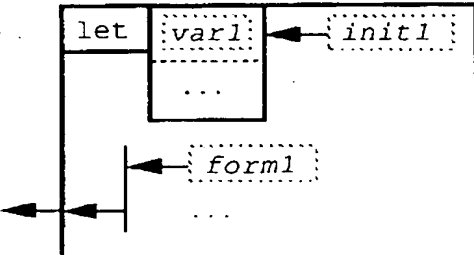
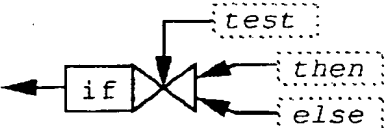
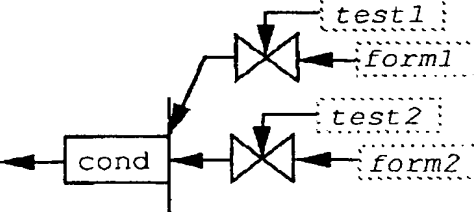

(図22)

212 変換キーの処理(その1)

【図24】

(図24)

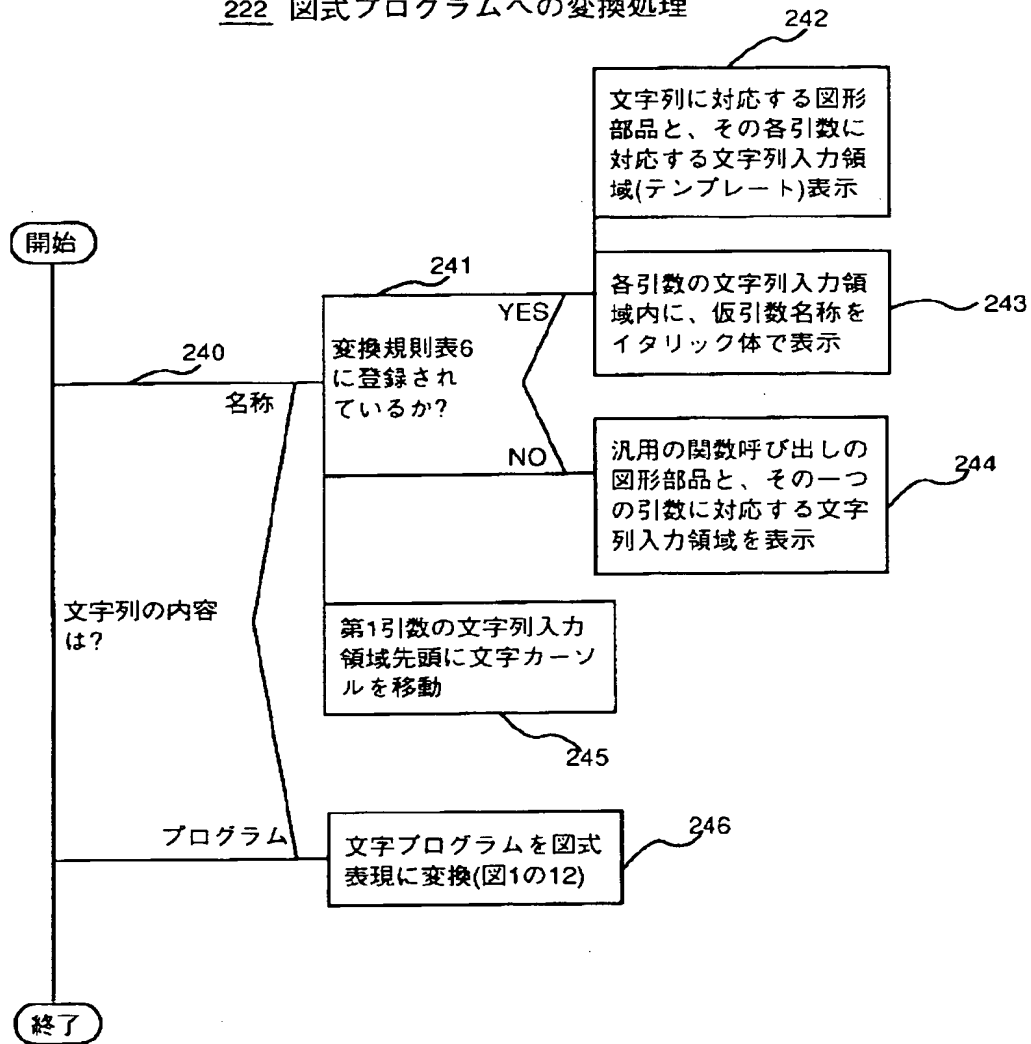
16 図式プログラム部品表

No	名称	文字表現の構文	図式プログラム部品
1	一般の関数	(<関数名> <引数1> <引数2>...)	
2	let	(let ((var1 init1) (var2 init2) ...) form1 form2 ...)	
3	if	(if test then [else])	
4	cond	(cond ((test1 form1) (test2 form2) ...))	
-
20	rem	(rem number divisor)	
-

【図25】

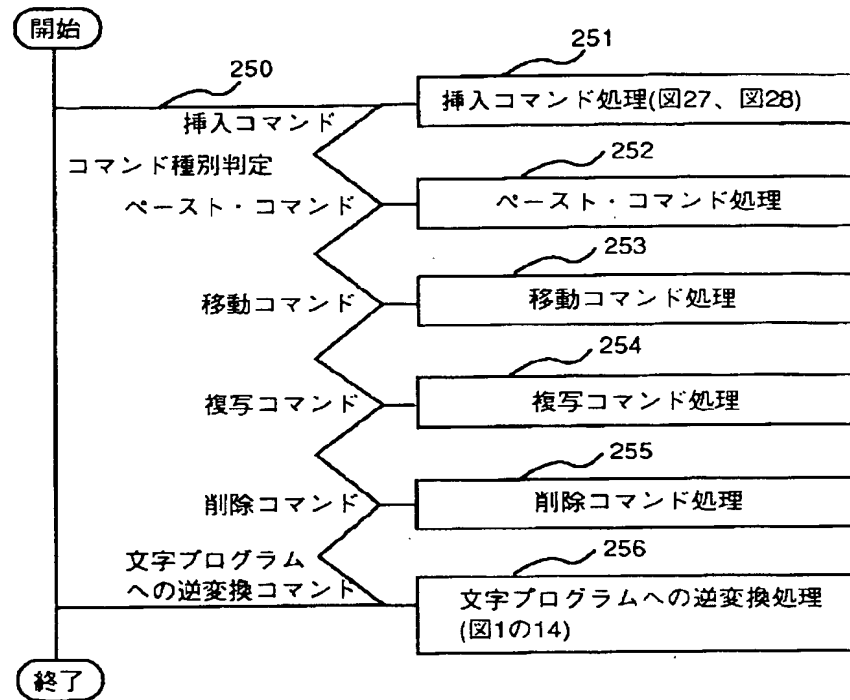
(図25)

222 図式プログラムへの変換処理



【図26】

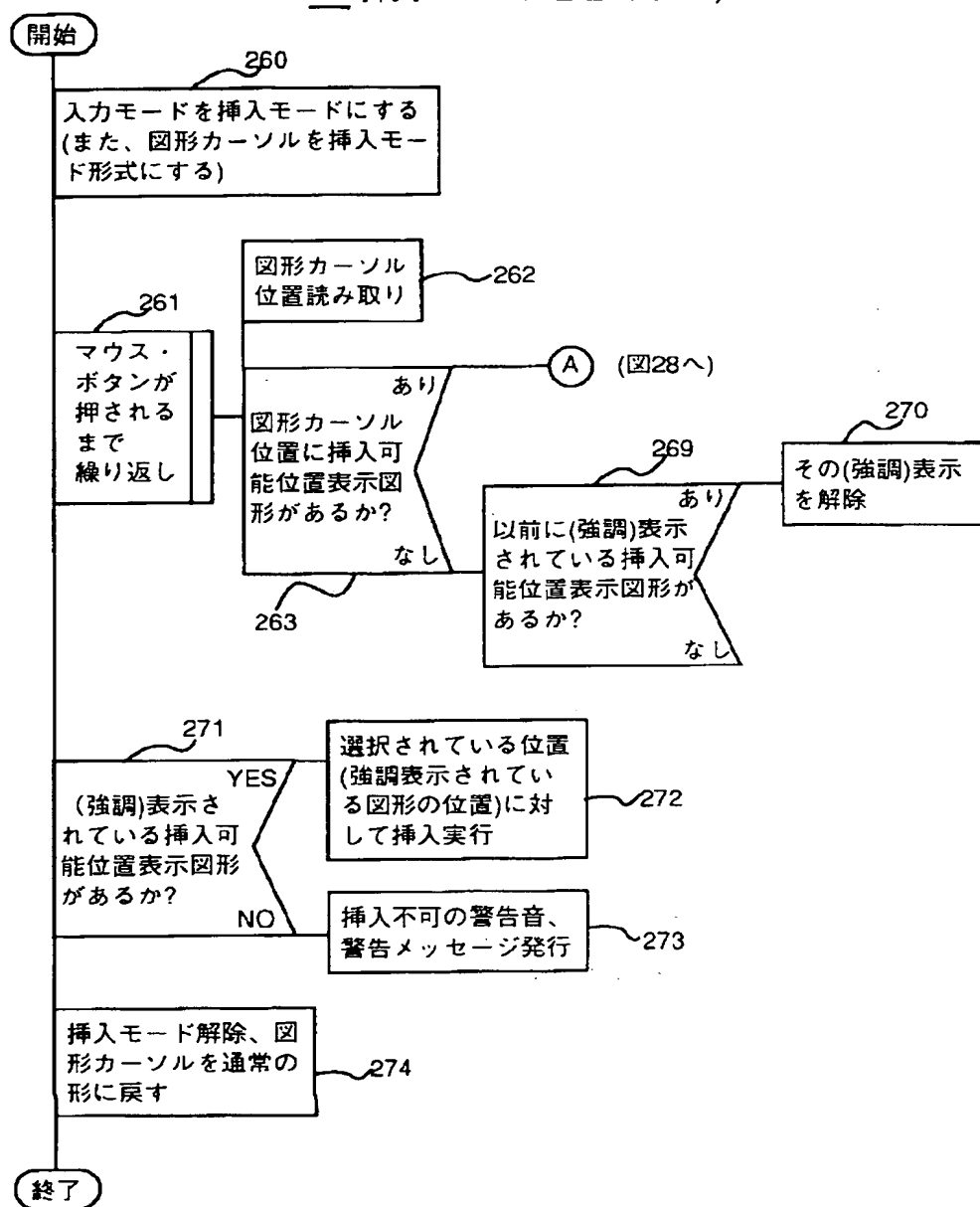
(図26)

204 コマンド処理

【図27】

(図27)

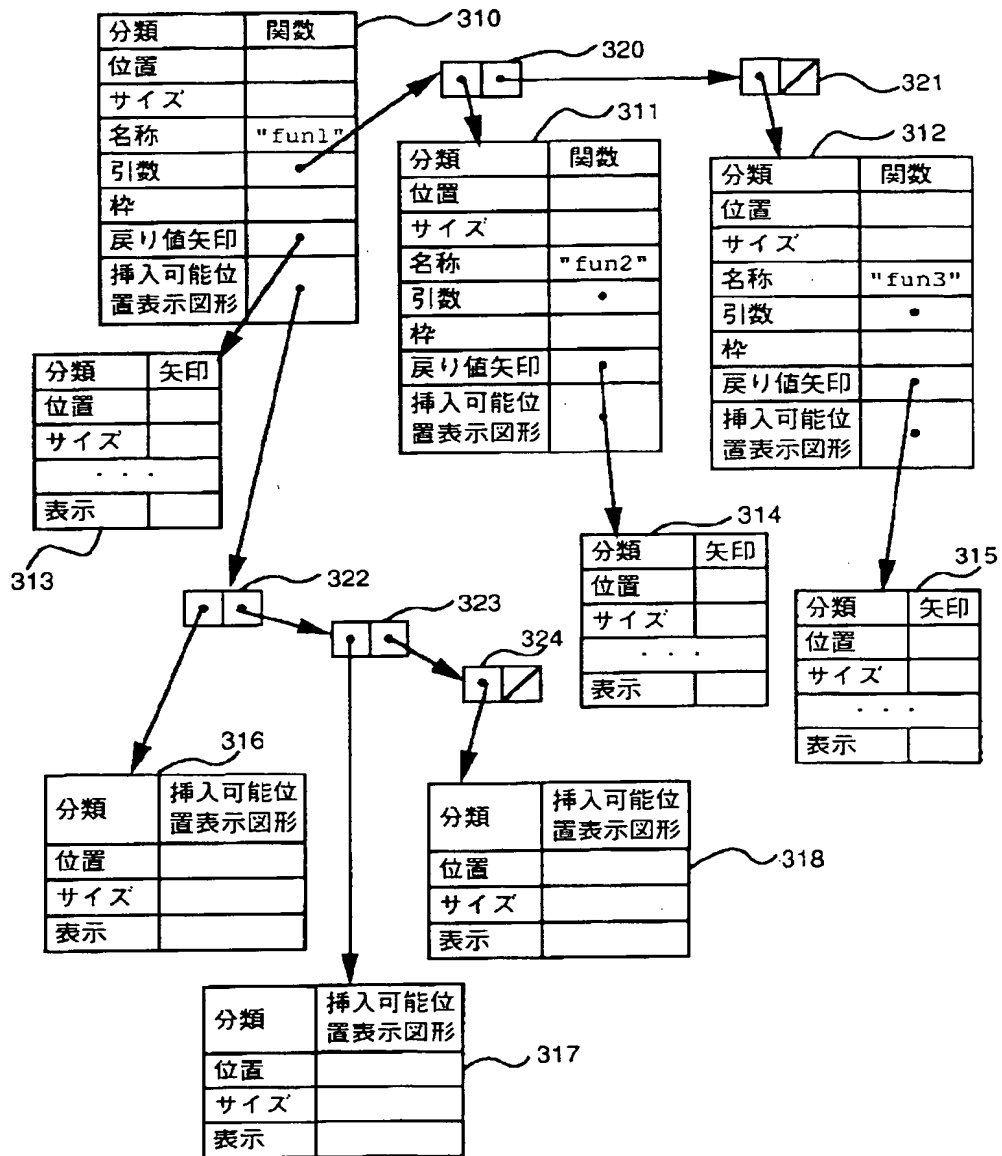
251 挿入コマンド処理 (その1)



【図30】

(図30)


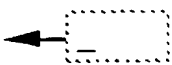
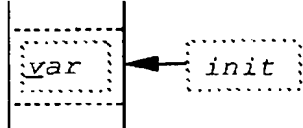
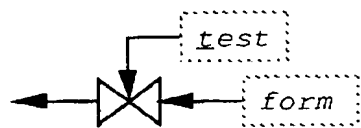
536 図式プログラムの内部データの例



【図31】

(図31)

537 文字列入力領域挿入位置と挿入される部品との対応表

No	挿入位置		挿入可能な図形プログラム部品	参考図
	図形部品	スロット名称		
1	関数	戻り値矢印		図17
2	一般の関数	引数		図18
3	let	変数リスト		図19
4	cond	節 (条件部/実行部ペア)		図20
	